AWS Whitepaper

Big Data Analytics Options on AWS

Big Data Analytics Options on AWS: AWS Whitepaper

Table of Contents

This whitepaper is for historical reference only. Some content might be outdated and some links might not be available.

Big Data Analytics Options on AWS

Publication date: **July 26, 2021** (*Document revisions*)

This whitepaper helps architects, data scientists, and developers understand the big data analytics options available in the Amazon Web Services (AWS) Cloud. It provides an overview of services, including:

- Ideal usage patterns
- Cost model
- Performance
- Durability and availability
- Scalability and elasticity
- Interfaces
- Anti-patterns

This paper concludes with scenarios that showcase the analytics options used, as well as additional resources for getting started with big data analytics on AWS.

Introduction

As the world becomes more digital, the amount of data created and collected constantly grows and accelerates. Analysis of this ever-growing data becomes a challenge with traditional analytical tools. Innovation is required to bridge the gap between generated data and data that can be analyzed effectively.

Big data tools and technologies offer opportunities to analyze data efficiently so you can better understand customer preferences, gain a competitive advantage in the marketplace, and grow your business. Data management architectures have evolved from the traditional data warehousing model to more complex architectures that address more requirements, such as real-time and batch processing, structured and unstructured data, high velocity transactions, and so on.

AWS provides a broad platform of managed services to help you build, secure, and seamlessly scale end-to-end big data applications quickly and with ease. Whether your applications require real-time streaming or batch data processing, AWS provides the infrastructure and tools to tackle your next big data project. There is no hardware to procure, no infrastructure to maintain and scale—only what you need to collect, store, process, and analyze big data. AWS has a system of analytical solutions specifically designed to handle this growing amount of data and provide insight into your business.

The AWS advantage in big data analytics

Analyzing large datasets requires significant compute capacity that can vary in size, based on the amount of input data and the type of analysis. This characteristic of big data workloads is ideally suited to the pay-as-you-go cloud computing model, where applications can easily scale up and down based on demand. As requirements change, you can easily resize your environment (horizontally or vertically) on AWS to meet your needs, without having to wait for additional hardware or over-investing to provision enough capacity.

For mission-critical applications on a more traditional infrastructure, system designers have no choice but to over-provision, because a surge in additional data due to an increase in business needs must be something the system can handle. By contrast, on AWS, you can provision more capacity and compute in a matter of minutes, meaning that your big data applications grow and shrink as demand dictates, and your system runs as close to optimal efficiency as possible.

In addition, you get flexible computing on a global infrastructure with access to the many different geographic Regions that AWS offers, along with the ability to use other scalable services that augment to build sophisticated big data applications. These other services include:

- Amazon Simple Storage Service (Amazon S3) to store data
- AWS Glue to orchestrate jobs to move and transform the data easily
- AWS IoT, which lets connected devices interact with cloud applications and other connected devices

As the amount of data being generated continues to grow, AWS has many options to get that data to the cloud, including secure devices like AWS Snow Family to accelerate petabyte-scale data transfers, delivery streams with Amazon Data Firehose to load streaming data continuously, migrating databases using AWS Database Migration Service, and scalable private connections through AWS Direct Connect.

As mobile continues to rapidly grow in usage, you can use the suite of services within the AWS Mobile Hub to collect and measure app usage and data, or export that data to another service for further custom analysis.

These capabilities of AWS make it an ideal fit for solving big data problems, and many customers have implemented successful big data analytics workloads on AWS. For more information about case studies, see Big Data Customer Success Stories.

The following services for collecting, processing, storing, and analyzing big data are described in order:

- Amazon Kinesis
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- AWS Lambda
- Amazon Elastic Map Reduce (Amazon EMR)
- AWS Glue
- AWS Lake Formation
- Amazon Machine Learning
- Amazon DynamoDB
- Amazon Redshift
- Amazon OpenSearch Service (OpenSearch Service)
- QuickSight
- Amazon Compute Services (Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS) are available for self-managed big data applications.)
- Amazon Athena

Amazon Kinesis

Amazon Kinesis is a platform for streaming data on AWS that makes it easy to load and analyze streaming data. Amazon Kinesis also enables you to build custom streaming data applications for specialized needs. With Kinesis, you can ingest real-time data such as application logs, website clickstreams, Internet of Things (IoT) telemetry data, and more into your databases, data lakes, and data warehouses, or build your own real-time applications using this data. Amazon Kinesis enables you to process and analyze data as it arrives and respond in real-time instead of having to wait until all your data is collected before the processing can begin.

Currently there are four pieces of the Kinesis platform that can be utilized based on your use case:

- Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data.

- Amazon Kinesis Video Streams enables you to build custom applications that process or analyze streaming video.
- Amazon Data Firehose enables you to deliver real-time streaming data to AWS destinations such as Amazon S3, Amazon Redshift, OpenSearch Service, and Splunk.
- Amazon Managed Service for Apache Flink enables you to process and analyze streaming data with standard SQL or with Java (managed Apache Flink).

Kinesis Data Streams and Kinesis Video Streams enable you to build custom applications that process or analyze streaming data in real time. Kinesis Data Streams can continuously capture and store terabytes of data per hour from hundreds of thousands of sources, such as website clickstreams, financial transactions, social media feeds, IT logs, and location-tracking events. Kinesis Video Streams can continuously capture video data from smartphones, security cameras, drones, satellites, dashcams, and other edge devices.

With the Amazon Kinesis Client Library (KCL), you can build Amazon Kinesis applications and use streaming data to power real-time dashboards, generate alerts, and implement dynamic pricing and advertising. You can also emit data from Kinesis Data Streams and Kinesis Video Streams to other AWS services such as Amazon S3, Amazon Redshift, Amazon EMR, and AWS Lambda.

Provision the level of input and output required for your data stream, in blocks of one megabyte per second (MB/sec), using the AWS Management Console, API, or SDKs. The size of your stream can be adjusted up or down at any time without restarting the stream and without any impact on the data sources pushing data to the stream. Within seconds, data put into a stream is available for analysis.

With Amazon Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Firehose and it automatically delivers the data to the AWS destination or third party (Splunk) that you specified. You can also configure Firehose to transform your data before data delivery. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security.

Amazon Managed Service for Apache Flink is the easiest way to process and analyze real-time, streaming data. With Managed Service for Apache Flink, you just use standard SQL or Java (Flink) to process your data streams, so you don’t have to learn any new programming languages. Simply point Managed Service for Apache Flink at an incoming data stream, write your SQL queries, and specify where you want to load the results. Managed Service for Apache Flink takes care of

running your SQL queries continuously on data while it's in transit and sending the results to the destinations.

For complex data processing applications, Amazon Managed Service for Apache Flink provides an option use open-source libraries such as Apache Flink, Apache Beam, AWS SDK, and AWS service integrations. It includes more than ten connectors from Apache Flink, and gives you the ability to build custom integrations. It's also compatible with the AWS Glue Schema Registry, a serverless feature of AWS Glue that enables you to validate and control the evolution of streaming data using registered Apache Avro schemas.

You can use Apache Flink in Amazon Managed Service for Apache Flink to build applications whose processed records affect the results exactly once, referred to as exactly once processing. This means that even in the case of an application disruption, like internal service maintenance or user-initiated application update, the service will ensure that all data is processed and there is no duplicate data. The service stores previous and in-progress computations, or *state*, in running application storage. This enables you to compare real-time and past results over any time period, and provides fast recovery during application disruptions.

The subsequent sections focus primarily on Amazon Kinesis Data Streams.

Ideal usage patterns

Amazon Kinesis Data Steams is useful wherever there is a need to move data rapidly off producers (data sources) and continuously process it. That processing can be to transform the data before emitting into another data store, drive real-time metrics and analytics, or derive and aggregate multiple streams into more complex streams, or downstream processing. The following are typical scenarios for using Kinesis Data Streams for analytics:

- **Real-time data analytics** – Kinesis Data Streams enables real-time data analytics on streaming data, such as analyzing website clickstream data and customer engagement analytics.
- **Log and data feed intake and processing** – With Kinesis Data Streams, you can have producers push data directly into an Amazon Kinesis stream. For example, you can submit system and application logs to Kinesis Data Streams and access the stream for processing within seconds. This prevents the log data from being lost if the front-end or application server fails, and reduces local log storage on the source. Kinesis Data Streams provides accelerated data intake because you are not batching up the data on the servers before you submit it for intake.
- **Real-time metrics and reporting** – You can use data ingested into Kinesis Data Streams for extracting metrics and generating KPIs to power reports and dashboards at real-time speeds.

This enables data-processing application logic to work on data as it is streaming in continuously, rather than waiting for data batches to arrive.

Cost model

Amazon Kinesis Data Streams has simple pay-as-you-go pricing, with no upfront costs or minimum fees, and you pay only for the resources you consume. An Amazon Kinesis stream is made up of one or more shards. Each shard gives you a capacity of five read transactions per second, up to a maximum total of 2 MB of data read per second. Each shard can support up to 1,000 write transactions per second, and up to a maximum total of 1 MB data written per second.

The data capacity of your stream is a function of the number of shards that you specify for the stream. The total capacity of the stream is the sum of the capacity of each shard. There are two components to pricing:

- Primary pricing includes an hourly charge per shard and a charge for each one million PUT transactions.
- Pricing for optional components for extended retention and enhanced fan-out.

For more information, see Amazon Kinesis Data Streams Pricing. Applications that run on Amazon EC2 and process Amazon Kinesis streams also incur standard Amazon EC2 costs.

Performance

Amazon Kinesis Data Streams enables you to choose the throughput capacity you require in terms of shards. With each shard in an Amazon Kinesis stream, you can capture up to 1 megabyte per second of data at 1,000 write transactions per second. Your Amazon Kinesis applications can read data from each shard at up to 2 megabytes per second. You can provision as many shards as you need to get the throughput capacity you want; for example, a one gigabyte per second data stream would require 1024 shards.

Additionally, there is a new feature. Enhanced fan-out enables developers to scale up the number of stream consumers (applications reading data from a stream in real-time) by offering each stream consumer their own read throughput. Developers can register stream consumers to use enhanced fan-out and receive their own 2MB/sec pipe of read throughput per shard. This throughput automatically scales with the number of shards in a stream.

Durability and availability

Amazon Kinesis Data Streams synchronously replicates data across three Availability Zones in an AWS Region, providing high availability and data durability.

Additionally, you can store a cursor in Amazon DynamoDB to durably track what has been read from an Amazon Kinesis stream. In the event that your application fails in the middle of reading data from the stream, you can restart your application and use the cursor to pick up from the exact spot where the failed application left off.

Scalability and elasticity

You can increase or decrease the capacity of the stream at any time according to your business or operational needs, without any interruption to ongoing stream processing. By using API calls or development tools, you can automate scaling of your Amazon Kinesis Data Streams environment to meet demand and ensure you only pay for what you need.

Interfaces

There are two interfaces to Kinesis Data Streams:

- Input which is used by data producers to put data into Kinesis Data Streams
- Output to process and analyze data that comes in

Producers can write data using the Amazon Kinesis PUT API, an AWS Software Development Kit (SDK) or toolkit abstraction, the Amazon Kinesis Producer Library (KPL), or the Amazon Kinesis Agent.

For processing data that has already been put into an Amazon Kinesis stream, there are client libraries provided to build and operate real-time streaming data processing applications. The KCL acts as an intermediary between Amazon Kinesis Data Streams and your business applications which contain the specific processing logic. There is also integration to read from an Amazon Kinesis stream into Apache Spark Streaming running on Amazon EMR.

Anti-patterns

Amazon Kinesis Data Streams has the following anti-patterns:

- **Small scale consistent throughput** – Even though Kinesis Data Streams works for streaming data at 200 KB per second or less, it is designed and optimized for larger data throughputs.
- **Long-term data storage and analytics** – Kinesis Data Streams is not suited for long-term data storage. By default, data is retained for 24 hours, and you can extend the retention period by up to 365 days.

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Amazon MSK is a fully managed service that makes it easy for you to build and run applications that use Apache Kafka to process streaming data. Apache Kafka is an open-source platform for building real-time streaming data pipelines and applications. With Amazon MSK, you can use native Apache Kafka APIs to populate data lakes, stream changes to and from databases, and power machine learning and analytics applications.

Apache Kafka clusters are challenging to set up, scale, and manage in production. When you run Apache Kafka on your own, you need to provision servers, configure Apache Kafka manually, replace servers when they fail, orchestrate server patches and upgrades, architect the cluster for high availability, ensure data is durably stored and secured, set up monitoring and alarms, and carefully plan scaling events to support load changes. Amazon MSK makes it easy for you to build and run production applications on Apache Kafka without needing Apache Kafka infrastructure management expertise.

With a few clicks in the Amazon MSK console (sign-in required), you can create a fully managed Apache Kafka cluster that follows Apache Kafka's deployment best practices, or you can create your own cluster using your own custom configuration. After you create your desired configuration, Amazon MSK automatically provisions, configures, and manages the operations of your Apache Kafka cluster and Apache ZooKeeper nodes.

An Amazon MSK cluster is the primary Amazon MSK resource that you can create in your account.

Following are the primary components that work together in MSK:

- **Broker nodes** — When creating an Amazon MSK cluster, you specify how many broker nodes you want Amazon MSK to create in each Availability Zone. Each Availability Zone has its own virtual private cloud (VPC) subnet.
- **ZooKeeper nodes** — Amazon MSK also creates the Apache ZooKeeper nodes for you. Apache ZooKeeper is an open-source server that enables highly reliable distributed coordination.

- **Producers, consumers, and topic creators** — Amazon MSK enables you to use Apache Kafka data-plane operations to create topics and to produce and consume data.
- **Cluster operations** — You can use the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the APIs in the SDK to perform control-plane operations. For example, you can create or delete an Amazon MSK cluster, list all the clusters in an account, view the properties of a cluster, and update the number and type of brokers in a cluster.

Amazon MSK detects and automatically recovers from the most common failure scenarios for clusters, so that your producer and consumer applications can continue their write and read operations with minimal impact. When Amazon MSK detects a broker failure, it mitigates the failure or replaces the unhealthy or unreachable broker with a new one. In addition, where possible, it reuses the storage from the older broker to reduce the data that Apache Kafka needs to replicate. Your availability impact is limited to the time required for Amazon MSK to complete the detection and recovery. After a recovery, your producer and consumer apps can continue to communicate with the same broker IP addresses that they used before the failure.

Ideal usage patterns

The AWS Streaming Data Solution for Amazon MSK enables you to capture, store, process, and deliver real-time streaming data. This service helps you address real-time streaming use cases; for example:

- Capture high volume application log files
- Analyze website clickstreams
- Process database event streams
- Track financial transactions
- Aggregate social media feeds
- Collect IT log files
- Continuously deliver to a data lake

Cost model

Amazon MSK has a simple “pay only for what you use” model. There are no minimum fees or upfront commitments. There are charges for the time your broker instances run, the storage used monthly, and standard data transfer fees for data in and out of the cluster. Apache Kafka broker

instance usage is billed on an hourly basis (billed at one second resolution), with varying fees depending on the size of the Apache Kafka broker instance and active brokers in your Amazon MSK clusters. Amazon MSK charges for the amount of storage you provision in the cluster. This is calculated by adding up the GB per broker each hour and dividing by the number of hours in the month, resulting in a value in "GB-Month."

There are no additional charges for data transfer between brokers or between Apache ZooKeeper nodes and brokers. Standard AWS data transfer charges are charged for data transferred in and out of Amazon MSK clusters. See Amazon MSK pricing for further details.

Performance

Amazon MSK allows you to choose the right type and number of brokers for your cluster. You can size your cluster based on your ingestion rate, hours of retention and data output rates. The number of partitions per broker is affected by use case and configuration. For more information about the different broker types, see Broker types.

Durability and availability

Use the following recommendations so that your MSK cluster can be highly available during an update (such as when you're updating the broker type or Apache Kafka version, for example) or when Amazon MSK is replacing a broker.

- Ensure that the replication factor (RF) is at least two for two-AZ clusters and at least three for three-AZ clusters. An RF of one can lead to offline partitions during a rolling update.
- Set minimum in-sync replicas (minISR) to at most RF-1. A minISR that is equal to the RF can prevent producing to the cluster during a rolling update. A minISR of two allows three-way replicated topics to be available when one replica is offline.
- Ensure client connection strings include multiple brokers. Having multiple brokers in a client's connection string allows for failover when a specific broker is offline for an update.

Scalability and elasticity

You can increase the capacity of the cluster at any time according to your business or operational needs. You can use Amazon MSK operation to increase the number of brokers in your MSK cluster. To expand a cluster, make sure that it is in the `ACTIVE` state.

You can also increase storage per broker. You can increase the amount of EBS storage per broker, but you can't decrease the storage. Storage volumes remain available during this scaling-up operation. You can use:

- **Automatic scaling** — You can configure Amazon Managed Streaming for Apache Kafka to automatically expand your cluster's storage in response to increased usage using Application Auto Scaling policies. Your automatic scaling policy sets the target disk utilization and the maximum scaling capacity.
- **Manual scaling** — To increase storage, wait for the cluster to be in the `ACTIVE` state. Storage scaling has a cooldown period of at least six hours between events. Even though the operation makes additional storage available right away, the service performs optimizations on your cluster that can take up to 24 hours or more. The duration of these optimizations is proportional to your storage size.

Interfaces

Amazon MSK has deep AWS service integrations with Amazon EMR, AWS Lambda, Amazon Managed Service for Apache Flink, and AWS Glue Streaming ETL. It also works with Kafka Connect, Mirror Maker, Kafka Streams, and a number of 3rd party frameworks like Apache Spark, Apache Storm, and so on. The producer side APIs add messages to the cluster for a topic. The consumer side APIs get messages for a topic as a stream of messages.

Anti-patterns

Amazon MSK has the following anti-patterns:

- **Ad hoc queries** — MSK is a stream of unbounded data. It is not used for ad hoc queries.
- **Long-term data storage and analytics** — MSK is not suited for long-term data storage.

AWS Lambda

AWS Lambda enables you to run code without provisioning or managing servers. You pay only for the compute time you consume – there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service – all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Ideal usage patterns

AWS Lambda enables you to run code in response to triggers such as changes in data, shifts in system state, or actions by users. Lambda can be directly triggered by AWS services such as Amazon S3, DynamoDB, Amazon Kinesis Data Streams, Amazon Simple Notification Service (Amazon SNS), and Amazon CloudWatch, enabling you to build a variety of real-time data processing systems:

- **Real-time file processing –** You can trigger Lambda to invoke a process where a file has been uploaded to Amazon S3 or modified. For example, to change an image from color to gray scale after it has been uploaded to Amazon S3.
- **Real-time stream processing** – You can use Kinesis Data Streams and Lambda to process streaming data for click stream analysis, log filtering, and social media analysis.
- **Extract, transform, load (ETL)** – You can use Lambda to run code that transforms data and loads that data into one data repository to another.
- **Replace Cron** – Use schedule expressions to run a Lambda function at regular intervals as a cheaper and more available solution than running cron on an EC2 instance.
- **Process AWS Events** – Many other services, such as AWS CloudTrail, can act as event sources simply by logging to Amazon S3 and using S3 bucket notifications to trigger Lambda functions.

Cost model

With AWS Lambda you only pay for what you use. You are charged based on the number of requests for your functions and the time your code runs. The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month. You are charged $0.20 per 1 million requests thereafter ($0.0000002 per request). Additionally, the duration of your code running is priced in relation to memory allocated. You are charged $0.00001667 for every GB-second used.

See AWS Lambda Pricing for more details.

Performance

After deploying your code into Lambda for the first time, your functions are typically ready to call within seconds of upload. Lambda is designed to process events within milliseconds. Latency will be higher immediately after a Lambda function is created, updated, or if it has not been used

recently. To improve performance, Lambda may choose to retain an instance of your function and reuse it to serve a subsequent request, rather than creating a new copy.

The Lambda provisioned concurrency feature provides customers greater control over performance of their serverless applications at any scale. Functions using provisioned concurrency run with consistent start-up latency, making them ideal for building interactive mobile or web backends, latency sensitive microservices, and synchronously invoked APIs. With provisioned concurrency, functions can instantaneously serve a burst of traffic with consistent start-up latency for every invoke up to the specified scale.

To learn more about how Lambda reuses function instances, see Getting started with Lambda. Your code should not assume that this reuse will always happen.

Durability and availability

AWS Lambda is designed to use replication and redundancy to provide high availability for both the service itself and for the Lambda functions it operates. There are no maintenance windows or scheduled downtimes for either. On failure, Lambda functions being invoked synchronously respond with an exception. Lambda functions being invoked asynchronously are retried at least three times, after which the event may be rejected.

Scalability and elasticity

There is no limit on the number of Lambda functions that you can run. However, Lambda has a default safety throttle of 1,000 concurrent runs per account per Region. A member of the AWS support team can increase this limit. Lambda is designed to scale automatically on your behalf. There are no fundamental limits to scaling a function. Lambda dynamically allocates capacity to match the rate of incoming events.

Interfaces

Lambda functions can be managed in a variety of ways. You can easily list, delete, update, and monitor your Lambda functions using the dashboard in the Lambda console. You also can use the AWS CLI and AWS SDK to manage your Lambda functions.

You can trigger a Lambda function from an AWS event, such as Amazon S3 bucket notifications, Amazon DynamoDB Streams, Amazon CloudWatch logs, Amazon Simple Email Service (Amazon SES), Amazon Kinesis Data Streams, Amazon SNS, Amazon Cognito, and more. Any API call in any service that supports AWS CloudTrail can be processed as an event in Lambda by responding to CloudTrail audit logs. For more information about event sources, see Lambda Event Sources.

AWS Lambda supports code written in Node.js (JavaScript), Python, Java (Java 8 compatible), C# (.NET Core), Go, PowerShell and Ruby. Your code can include existing libraries, even native ones. See AWS documentation on using Node.js, Python, Java, C#, Go, PowerShell, and Ruby.

You can package your Lambda function code and dependencies as a container image, using tools such as the Docker CLI. You can then upload the image to your container registry hosted on Amazon ECR. Note that you must create the Lambda function from the same account as the container registry in Amazon ECR.

Anti-patterns

AWS Lambda has the following anti-patterns:

- **Long running application** — Each Lambda function must complete within 900 seconds. For long running applications that may require jobs to run longer than fifteen minutes, Amazon EC2, Amazon EKS, or Amazon ECS is recommended. Alternately, you can use Step Functions.
- **Dynamic websites** — While it is possible to run a static website with AWS Lambda, running a highly dynamic and large volume website can be performance prohibitive. Utilizing Amazon EC2, Amazon EKS, or Amazon ECS and AWS CloudFormation is a recommended use-case.
- **Stateful applications** — Lambda code must be written in a "stateless" style, meaning it should assume there is no affinity to the underlying compute infrastructure. Local file system access, child processes, and similar artifacts may not extend beyond the lifetime of the request, and any persistent state should be stored in Amazon S3, DynamoDB, or another internet-available storage service.

Amazon EMR

Amazon EMR is the industry-leading cloud big data platform for processing vast amounts of data using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi, and Presto. Amazon EMR makes it easy to set up, operate, and scale your big data environments by automating time-consuming tasks like provisioning capacity and tuning clusters. With EMR you can run petabyte-scale analysis at less than half of the cost of traditional on-premises solutions and over 3x faster than standard Apache Spark. You can run workloads on Amazon EC2 instances, on Amazon EKS clusters, or on-premises using EMR on AWS Outposts.

Amazon EMR does all the work involved with provisioning, managing, and maintaining the infrastructure and software of a Hadoop cluster. Amazon EMR now supports Amazon EC2 M6g instances to deliver the best price performance for cloud workloads. Amazon EC2 M6g instances

are powered by AWS Graviton2 processors that are custom designed by AWS, utilizing 64-bit Arm Neoverse cores. Amazon EMR provides up to 35% lower cost and up to 15% improved performance for Spark workloads on Graviton2-based instances versus previous generation instances.

Ideal usage patterns

Amazon EMR's flexible framework reduces large processing problems and data sets into smaller jobs and distributes them across many compute nodes in a Hadoop cluster. This capability lends itself to many usage patterns with big data analytics. Here are a few examples:

- Log processing and analytics
- Large ETL data movement
- Risk modeling and threat analytics
- Ad targeting and click stream analytics
- Genomics
- Predictive analytics
- Ad hoc data mining and analytics

For more information, see the documentation for Amazon EMR.

Cost model

With Amazon EMR, you can launch a persistent cluster that stays up indefinitely, or a temporary cluster that ends after the analysis is complete. In either scenario, you pay only for the hours the cluster is up.

Amazon EMR supports a variety of Amazon EC2 instance types (standard, high CPU, high memory, high I/O, and so on) and all Amazon EC2 pricing options (On-Demand, Reserved, and Spot). When you launch an Amazon EMR cluster (also called a "job flow"), you choose how many and what type of Amazon EC2 instances to provision. The Amazon EMR price is in addition to the Amazon EC2 price.

For more information, see Amazon EMR pricing.

Performance

Amazon EMR performance is driven by the type of EC2 instances on which you choose to run your cluster, and how many you chose to run your analytics. You should choose an instance type suitable

for your processing requirements, with sufficient memory, storage, and processing power. With the introduction of Graviton2 instances, you can see improved performance of up to 15% relative to equivalent previous generation instances. For more information about EC2 instance specifications, see Amazon EC2 Instance Types.

An important consideration when you create an EMR cluster is how you configure Amazon EC2 instances. EC2 instances in an EMR cluster are organized into node types.

The node types in Amazon EMR are as follows:

- **Primary node** — A node that manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing. The primary node tracks the status of tasks and monitors the health of the cluster. Every cluster has a primary node, and it's possible to create a single-node cluster with only the primary node.
- **Core node** — A node with software components that run tasks and store data in the Hadoop Distributed File System (HDFS) on your cluster. Multi-node clusters have at least one core node.
- **Task node** — A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.

Durability and availability

By default, Amazon EMR is fault tolerant for core node failures and continues job execution if a dependent node goes down. Amazon EMR will also provision a new node when a core node fails. However, Amazon EMR will not replace nodes if all nodes in the cluster are lost.

You can monitor the health of nodes and replace failed nodes with Amazon CloudWatch. When you launch an Amazon EMR cluster, you can choose to have one or three primary nodes in your cluster. Launching a cluster with three primary nodes is only supported by Amazon EMR version 5.23.0 and later. EMR can take advantage of EC2 placement groups to ensure primary nodes are placed on distinct underlying hardware to further improve cluster availability.

For more information, see EMR integration with EC2 placement groups.

Scalability and elasticity

With Amazon EMR, it's easy to resize a running cluster. You can add core nodes which hold the HDFS at any time to increase your processing power and increase the HDFS storage capacity (and throughput). Additionally, you can use Amazon S3 natively, or using EMRFS along with or instead

of local HDFS, which enables you to decouple your memory and compute from your storage providing greater flexibility and cost efficiency.

You can also add and remove task nodes at any time which can process Hadoop jobs, but do not maintain HDFS. Some customers add hundreds of instances to their clusters when their batch processing occurs, and remove the extra instances when processing completes. For example, you may not know how much data your clusters will be handling in six months, or you may have spiky processing needs.

With Amazon EMR, you don't need to guess your future requirements or provision for peak demand because you can easily add or remove capacity at any time.

You can add all new clusters of various sizes and remove them at any time with a few clicks in the console or by a programmatic API call.

Additionally, you can configure instance fleets for a cluster to choose a wide variety of provisioning options for EC2 instances. With instance fleets you can specify target capacities on On-Demand Instances, and Spot Instances within each fleet. Amazon EMR tries to provide the capacity you need with the best mix of capacity and price based on your selection of Availability Zones.

While a cluster is running, if Amazon EC2 reclaims a Spot Instance because of a price increase, or an instance fails, Amazon EMR tries to replace the instance with any of the instance types that you specify. This makes it easier to regain capacity if a node is lost for any reason.

Interfaces

Amazon EMR supports many tools on top of Hadoop that can be used for big data analytics and each has their own interfaces. Here is a brief summary of the most popular options:

Hive

Hive is an open source data warehouse and analytics package that runs on top of Hadoop. Hive is operated by Hive QL, a SQL-based language, which enables users to structure, summarize, and query data. Hive QL goes beyond standard SQL, adding first-class support for map/reduce functions and complex extensible user-defined data types like JSON and Thrift. This capability allows processing of complex and unstructured data sources such as text documents and log files.

Hive allows user extensions via user-defined functions written in Java. Amazon EMR has made numerous improvements to Hive, including direct integration with DynamoDB and Amazon S3. For example, with Amazon EMR you can load table partitions automatically from S3, you can write

data to tables in S3 without using temporary files, and you can access resources in S3, such as scripts for custom map and/or reduce operations and additional libraries.

For more information, see Apache Hive in the *Amazon EMR Release Guide*.

Pig

Pig is an open-source analytics package that runs on top of Hadoop. Pig is operated by Pig Latin, an SQL-like language which enables users to structure, summarize, and query data. Pig Latin also adds first-class support for map and reduce functions and complex extensible user-defined data types. This capability allows processing of complex and unstructured data sources such as text documents and log files.

Pig allows user extensions via user-defined functions written in Java. Amazon EMR has made numerous improvements to Pig, including the ability to use multiple file systems (normally, Pig can only access one remote file system), the ability to load customer JARs and scripts from S3 (such as "`REGISTER s3://my-bucket/piggybank.jar`"), and additional functionality for String and DateTime processing.

For more information, see Apache Pig in the *Amazon EMR Release Guide*.

Spark

Spark is an open-source data analytics engine built on Hadoop with the fundamentals for in-memory MapReduce. Spark provides additional speed for certain analytics and is the foundation for other power tools such as Shark (SQL driven data warehousing), Spark Streaming (streaming applications), GraphX (graph systems) and MLlib (machine learning).

EMR features Amazon EMR runtime for Apache Spark, a performance-optimized runtime environment for Apache Spark that is active by default on Amazon EMR clusters. Amazon EMR runtime for Apache Spark can be over 3x faster than clusters without the EMR runtime, and has 100% API compatibility with standard Apache Spark. This improved performance means your workloads run faster and saves you compute costs, without making any changes to your applications.

For more information, see Apache Spark on Amazon EMR .

HBase

HBase is an open-source, non-relational, distributed database modeled after Google's Bigtable. It was developed as part of Apache Software Foundation's Hadoop project and runs on top of

Hadoop Distributed File System (HDFS) to provide BigTable-like capabilities for Hadoop. HBase provides you a fault-tolerant, efficient way of storing large quantities of sparse data using column-based compression and storage. In addition, HBase provides fast lookup of data, because data is stored in-memory instead of on disk.

HBase is optimized for sequential write operations, and it is highly efficient for batch inserts, updates, and deletes. HBase works seamlessly with Hadoop, sharing its file system and serving as a direct input and output to Hadoop jobs. HBase also integrates with Apache Hive, enabling SQL-like queries over HBase tables, joins with Hive-based tables, and support for Java Database Connectivity (JDBC). With Amazon EMR, you can back up HBase to Amazon S3 (full or incremental, manual or automated) and you can restore from a previously created backup.

For more information, see Apache HBase in the *Amazon EMR Release Guide*.

Presto

Presto is an open-source distributed SQL query engine optimized for low-latency, ad hoc analysis of data. It supports the ANSI SQL standard, including complex queries, aggregations, joins, and window functions. Presto can process data from multiple data sources, including HDFS and Amazon S3.

Hudi

Apache Hudi is an open-source data management framework used to simplify incremental data processing and data pipeline development by providing record-level insert, update, upsert, and delete capabilities. *Upsert* refers to the ability to insert records into an existing dataset if they do not already exist or to update them if they do. By efficiently managing how data is laid out in S3, Hudi allows data to be ingested and updated in near-real-time. Hudi carefully maintains metadata of the actions performed on the dataset to help ensure that the actions are atomic and consistent. Hudi is integrated with Apache Spark, Apache Hive, and Presto. In Amazon EMR release versions 6.1.0 and later, Hudi is also integrated with Trino (PrestoSQL).

Kinesis Connector

The Kinesis Connector enables EMR to directly read and query data from Kinesis Data Streams. You can perform batch processing of Kinesis streams using existing Hadoop tools such as Hive, Pig, MapReduce, Hadoop Streaming, and Cascading. Some use cases enabled by this integration are:

- **Streaming log analysis** — You can analyze streaming web logs to generate a list of top ten error types every few minutes by Region, browser, and access domains.

- **Complex data processing workflows** — You can join Kinesis stream with data stored in Amazon S3, Dynamo DB tables, and HDFS. You can write queries that join clickstream data from Kinesis with advertising campaign information stored in a DynamoDB table to identify the most effective categories of ads that are displayed on particular websites.
- **Ad-hoc queries** — You can periodically load data from Kinesis into HDFS and make it available as a local Impala table for fast, interactive, analytic queries.

Other third-party tools

Amazon EMR also supports a variety of other popular applications and tools in the Hadoop system, such as R (statistics), Mahout (machine learning), Ganglia (monitoring), Accumulo (secure NoSQL database), Hue (user interface to analyze Hadoop data), HCatalog (table and storage management), and more.

Additionally, you can install your own software on top of Amazon EMR to help solve your business needs. AWS provides the ability to quickly move large amounts of data from S3 to HDFS, from HDFS to S3, and between S3 buckets using Amazon EMR's S3DistCp, an extension of the open source tool DistCp that uses MapReduce to efficiently move large amounts of data.

You can optionally use the EMR File System (EMRFS), an implementation of HDFS which allows Amazon EMR clusters to store data on S3. You can enable S3 server-side and client-side encryption.

EMR Studio

EMR Studio is an integrated development environment (IDE) that makes it easy for data scientists and data engineers to develop, visualize, and debug data engineering and data science applications written in R, Python, Scala, and PySpark.

EMR Studio provides fully managed Jupyter Notebooks, and tools like Spark UI and YARN Timeline Service to simplify debugging. Data scientists and analysts can install custom kernels and libraries, collaborate with peers using code repositories such as GitHub and BitBucket, or run parameterized notebooks as part of scheduled workflows using orchestration services like Apache Airflow or Amazon Managed Workflows for Apache Airflow. Administrators can set up EMR Studio such that analysts can run their applications on existing EMR clusters, or create new clusters using pre-defined AWS CloudFormation templates for EMR.

Anti-patterns

Amazon EMR has the following anti-pattern:

- **Small datasets** – Amazon EMR is built for massive parallel processing; if your data set is small enough to run quickly on a single machine, in a single thread, the added overhead to map and reduce jobs may not be worth it for small datasets that can easily be processed in memory on a single system. Amazon EMR or a relational database running on Amazon EMR may be a better option for workloads with stringent requirements.

AWS Glue

AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development. AWS Glue provides all of the capabilities needed for data integration. It uses both visual and code-based interfaces to make data integration easier.

Users can easily find and access data using the AWS Glue Data Catalog. Data engineers and ETL developers can visually create, run, and monitor ETL workflows with a few clicks in AWS Glue Studio. Data analysts and data scientists can use AWS Glue DataBrew to visually enrich, clean, and normalize data without writing code.

Ideal usage patterns

AWS Glue is designed to easily prepare data for extract, transform, and load (ETL) jobs. Using AWS Glue gives you the following benefits:

- **Data discovery**
  - Automatic schema discovery, using AWS Glue crawlers
  - Manage and enforce schemas for data streams with AWS Glue Schema Registry
- **Data Catalog**
  - The AWS Glue Data Catalog is a central repository to store structural and operational metadata for all your data assets. For a given dataset, you can store its table definition, physical location, add business relevant attributes, as well as track how this data has changed over time.
  - The AWS Glue Data Catalog is Apache Hive Metastore-compatible and is a drop-in replacement for the Apache Hive Metastore for Big Data applications running on Amazon EMR, and third-party applications such as Databricks.
- **Data transformation**

- Visually transform data with a drag and drop interface using AWS Glue Studio. AWS Glue automatically generates the reusable and portable code using familiar technology – Python (or Scala) and Spark.
- Serverless streaming ETL jobs in AWS Glue continuously consume data from streaming sources including Amazon Kinesis and Amazon MSK, clean and transform it in-flight, and make it available for analysis in seconds in your target data store.

- **Data replication**
  - AWS Glue Elastic Views enables you to create views over data stored in multiple types of AWS data stores, and materialize the views in a target data store of your choice by writing queries in PartiQL, an open-source SQL-compatible query language.
- **Data preparation**
  - Deduplicate and cleanse data with built-in machine learning. The FindMatches feature deduplicates and finds records that are imperfect matches of each other.
  - Normalize data without code using a visual interface. AWS Glue DataBrew provides an interactive, point-and-click visual interface for users like data analysts and data scientists to clean and normalize data without writing code. Choose from over 250 built-in transformations.
- **Integration**
  - Integration with data access services like Amazon Athena, Amazon EMR, and Amazon Redshift. Also with third parties.
- **Serverless**
  - No infrastructure to provision or manage.

Cost model

With AWS Glue jobs (crawler and ETL), AWS Glue DataBrew jobs, and AWS Glue Elastic Views, you pay an hourly rate, billed by the second with a 1-minute minimum billing duration. For the AWS Glue Data Catalog, you pay a simple monthly fee for storing and accessing the metadata. The first million objects stored are free, and the first million accesses are free. If you provision a development endpoint to interactively develop your ETL code, you pay an hourly rate, billed per minute. AWS Glue DataBrew interactive sessions bill for the total number of the sessions used. Each session is 30 minutes. A session is initiated when you open a DataBrew project.

See AWS Glue pricing for more details.

Performance

AWS Glue uses a scale-out Apache Spark environment to load your data into its destination. You can simply specify the number of Data Processing Units (DPUs) that you want to allocate to your ETL job. An AWS Glue ETL job requires a minimum of 2 DPUs. By default, AWS Glue allocates 10 DPUs to each ETL job. Additional DPUs can be added to increase the performance of your ETL job. Multiple jobs can be triggered in parallel or sequentially by triggering them on a job completion event. You can also trigger one or more AWS Glue jobs from an external source such as AWS Step Functions or Amazon Managed Workflows for Apache Airflow.

Durability and availability

AWS Glue connects to the data source of your preference, whether it is in an Amazon S3 file, an Amazon RDS table, or another set of data. As a result, all your data is stored and available as it pertains to that data stores durability characteristics. The AWS Glue service provides status of each job and pushes all notifications to Amazon CloudWatch events. You can setup SNS notifications using CloudWatch actions to be informed of job failures or completions.

Scalability and elasticity

AWS Glue provides a managed ETL service that runs on a Serverless Apache Spark environment. This enables you to focus on your ETL job and not worry about configuring and managing the underlying compute resources. AWS Glue works on top of the Apache Spark environment to provide a scale-out run environment for your data transformation jobs.

Interfaces

AWS Glue provides a number of ways to populate metadata into the AWS Glue Data Catalog. AWS Glue crawlers scan various data stores you own to automatically infer schemas and partition structure and populate the AWS Glue Data Catalog with corresponding table definitions and statistics. You can also schedule crawlers to run periodically so that your metadata is always up-to-date and in-sync with the underlying data.

Alternately, you can add and update table details manually by using the AWS Glue Console or by calling the API. You can also run Hive DDL statements via the Amazon Athena Console or a Hive client on an Amazon EMR cluster.

Finally, if you already have a persistent Apache Hive Metastore, you can perform a bulk import of that metadata into the AWS Glue Data Catalog by using the import script.

Anti-patterns

AWS Glue has the following anti-patterns:

- **Multiple ETL engines** – AWS Glue ETL jobs are Spark-based. If your use case requires you to use an engine other than Apache Spark or if you want to run a heterogeneous set of jobs that run on a variety of engines like Hive or Pig, Amazon EMR is a better choice.
- **Configurable Spark environment** – AWS Glue is a fully managed service. While you can change some configuration parameters in your cluster, if your use case requires extensive configuration changes, Amazon EMR or Amazon EMR on EKS is a better choice.

AWS Lake Formation

AWS Lake Formation is an integrated data lake service that makes it easy for you to ingest, clean, catalog, transform, and secure your data and make it available for analysis and machine learning. Lake Formation gives you a central console where you can discover data sources, set up transformation jobs to move data to an S3 data lake, remove duplicates and match records, catalog data for access by analytic tools, configure data access and security policies, and audit and control access from AWS analytic and machine learning services.

Lake Formation automatically manages access to the registered data in S3 via services including AWS Glue, Amazon Athena, Amazon Redshift, Amazon EMR Notebooks and Zeppelin notebooks with Apache Spark, and QuickSight to ensure compliance with your defined policies. If you've set up transformation jobs spanning AWS services, Lake Formation configures the flows, centralizes their orchestration, and lets you monitor the running of your jobs. With Lake Formation, you can configure and manage your data lake without manually integrating multiple underlying AWS services.

Ideal usage patterns

AWS Lake Formation helps you collect and catalog data from databases and object storage, move the data into your new S3 data lake, clean and classify your data using machine learning algorithms, and secure access to your sensitive data. Your users can access a centralized data catalog which describes available datasets and their appropriate usage. Using AWS Lake Formation gives you the following benefits:

- **Build data lakes quickly**

- Lake Formation blueprints simplify the deployment of ingestion workflow via a simple interface.
- Simply point Lake Formation at your data sources, and Lake Formation crawls those sources and moves the data into your new S3 data lake.
- Lake Formation has built-in machine learning to deduplicate and find matching records (two entries that refer to the same thing) to increase data quality.

- **Simplify security management**
  - Centrally define security, governance, and auditing policies in one place.
  - You can define security policy-based rules for your users and applications by role in Lake Formation, and integration with AWS Identity and Access Management (AWS IAM) authenticates those users and roles.
  - With tag-based access control, data stewards can define a tag ontology based on data classification, and grant access based on tags to IAM principals and SAML principals or groups.
  - Enforce encryption leveraging the encryption capabilities of S3 for data in your data lake.
  - Provides comprehensive audit logs with CloudTrail to monitor access and show compliance with centrally defined policies.
- **Provide self-service access to data**
  - Label your data with business metadata. Designate data owners, such as data stewards and business units, by adding a field in table properties as “custom attributes”.
  - Specify, grant, and revoke permissions on tables defined in the central Data Catalog. The same Data Catalog is available for multiple accounts, groups, and services.
  - Search for relevant data by name, contents, sensitivity, or other any other custom labels you have defined.
- **Integration**
  - Integration with data access services like Amazon Athena, Amazon EMR, Amazon Redshift, and Amazon QuickSight.
- **Serverless**
  - No infrastructure to provision or manage.

Cost model

There is no additional charge for using features in Lake Formation. Lake Formation helps you build and manage data lakes where your data in stored in S3. It builds on capabilities available in AWS

Glue and uses the AWS Glue Data Catalog, jobs, and crawlers. It also integrates with services like AWS CloudTrail, AWS IAM, Amazon CloudWatch, Amazon Athena, Amazon EMR, Amazon Redshift, and others.

Standard usage rates for these services will apply based on pricing for these services.

Performance

AWS Lake Formation uses AWS Glue as a transformation engine. Refer to the AWS Lake Formation: How it Works section of the Lake Formation Developer Guide.

Durability and availability

AWS Lake Formation leverage S3 as data lake storage, designed to provide 99.999999999% of data durability of objects over a given year. It creates and stores copies of all S3 objects across multiple systems. This means your data is available when you need it, and protected against failures, errors, and threats. The AWS Glue service is fully managed. It provides status for each job and pushes all notifications to Amazon CloudWatch Events. You can setup SNS notifications using CloudWatch actions to be informed of job failures or completions.

Scalability and elasticity

AWS Lake Formation uses AWS Glue, which provides a managed ETL service that runs on a Serverless Apache Spark environment. This enables you to focus on your ETL job and not worry about configuring and managing the underlying compute resources. AWS Glue works on top of the Apache Spark environment to provide a scale-out run environment for your data transformation jobs.

Interfaces

Lake Formation provides a console to manage your data lake locations, data catalog, access privileges, and deploy blueprints. It also provides APIs and a CLI to integrate Lake Formation functionality into your custom applications. Java and C++ SDKs are available to enable you to integrate your own data engines with Lake Formation. Lake Formation manages query access from AWS Glue, Athena, Amazon Redshift, EMR Notebooks, and Zeppelin notebooks for EMR with Apache Spark through a unified security model and permissions.

You can use third-party business applications like Tableau and Looker to connect to your AWS data sources through services like Athena or Amazon Redshift. Access to data is managed by the

underlying data catalog, so regardless of which application you use, you are assured that access to your data is governed and controlled.

Anti-patterns

AWS Lake Formation has the following anti-patterns:

- **Managing unstructured data** – AWS Lake Formation does not manage security for unstructured objects as document (PDF, Word), images, or videos.
- **Business catalog** – Lake Formation helps label your data with business metadata and designate data owners such as data stewards and business units by adding a field in table properties as "custom attributes". For advanced use cases, consider integrating a Partner solution from the AWS Marketplace.
- **Managing permission for open source and third-party big data components** — AWS Lake Formation is not integrated with Presto, HBase and other EMR components not specified in the previous **Interface** section of this chapter. It is also not integrated with third party applications such as Trino or Databricks.
- **Real-time analytics** – AWS Lake Formation is not integrated with Amazon Kinesis or OpenSearch Service. If you need to manage a real-time analytics pipeline or dashboard, you will have to manage privileges in a traditional way.

Amazon Machine Learning

AWS offers the broadest and deepest set of machine learning services and supporting cloud infrastructure, putting machine learning in the hands of every developer, data scientist, and expert practitioner. When you build an ML-based workload in AWS, you can choose from three different levels of ML services to balance speed-to-market with level of customization and ML skill level:

- Artificial Intelligence (AI) services
- ML services
- ML frameworks and infrastructure

The AI Services level provides fully managed services that enable you to quickly add ML capabilities to your workloads using API calls. This gives you the ability to build powerful, intelligent applications with capabilities such as computer vision, speech, natural language, chatbots,

predictions, and recommendations. Services at this level are based on pre-trained or automatically trained machine learning and deep learning models, so you don't need ML knowledge to use them.

You can use:

- Amazon Translate to translate or localize text content
- Amazon Polly for text-to-speech conversion
- Amazon Lex for building conversational chat bots
- Amazon Comprehend to extract insights and relationships from unstructured data
- Amazon Forecast to build accurate forecasting models
- Amazon Fraud Detector to identify potentially fraudulent online activities,
- Amazon CodeGuru to automate code reviews and to identify most expensive lines of code
- Amazon Textract to extract text and data from documents automatically
- Amazon Rekognition to add image and video analysis to your applications
- Amazon Kendra to reimagines enterprise search for your websites and applications
- Amazon Personalize for real-time personalized recommendations
- Amazon Transcribe to add speech to text capabilities to your applications

The ML Services level provides managed services and resources for machine learning to developers, data scientists, and researchers.

- Amazon SageMaker AI enables developers and data scientists to quickly and easily build, train, and deploy ML models at any scale.
- Amazon SageMaker AI Ground Truth helps you build highly accurate ML training datasets quickly.
- Amazon SageMaker AI Studio is the first integrated development environment for machine learning to build, train, and deploy ML models at scale.
- Amazon SageMaker AI Autopilot automatically builds, trains, and tunes the best ML models based on your data, while enabling you to maintain full control and visibility.
- Amazon SageMaker AI JumpStart helps you quickly and easily get started with ML.
- Amazon SageMaker AI Data Wrangler reduces the time it takes to aggregate and prepare data for ML from weeks to minutes.
- Amazon SageMaker AI Feature Store is a fully managed, purpose-built repository to store, update, retrieve, and share ML features.

- Amazon SageMaker AI Clarify provides ML developers with greater visibility into your training data and models so you can identify and limit bias and explain predictions.
- Amazon SageMaker AI Debugger optimizes ML models with real-time monitoring of training metrics and system resources.
- Amazon SageMaker AI's distributed training libraries automatically split large deep learning models and training datasets across AWS graphics processing unit (GPU) instances in a fraction of the time it takes to do manually.
- Amazon SageMaker AI Pipelines is the first purpose-built, easy-to-use continuous integration and continuous delivery (CI/CD) service for ML.
- Amazon SageMaker AI Neo enables developers to train ML models once, and then run them anywhere in the cloud or at the edge.

The ML Frameworks and Infrastructure level is intended for expert ML practitioners. These people are comfortable with designing their own tools and workflows to build, train, tune, and deploy models, and are accustomed to working at the framework and infrastructure level. In AWS, you can use open-source ML frameworks such as TensorFlow, PyTorch, and Apache MXNet. The Deep Learning AMI and Deep Learning Containers in this level have multiple ML frameworks preinstalled that are optimized for performance. This optimization means that they are always ready to be launched on powerful, ML-optimized compute infrastructure, such as Amazon EC2 P3 and P3dn instances, that provides a boost of speed and efficiency to ML workloads.

Amazon ML can create ML models based on data stored in S3, Amazon Redshift, or Amazon RDS. Built-in wizards guide you through the steps of interactively exploring your data to training the ML model, evaluate the model quality, and adjust outputs to align with business goals. After a model is ready, you can request predictions in batches, or using the low-latency real-time API.

Workloads often use services from multiple levels of the ML stack. Depending on the business use case, services and infrastructure from the different levels can be combined to satisfy multiple requirements and achieve multiple business goals. For example, you can use AI services for sentiment analysis of customer reviews on your retail website, and use managed ML services to build a custom model using your own data to predict future sales.

Ideal usage patterns

Amazon ML is ideal for discovering patterns in your data and using these patterns to create ML models that can generate predictions on new, unseen data points. For example, you can:

- **Enable applications to flag suspicious transactions** – Build an ML model that predicts whether a new transaction is legitimate or fraudulent.
- **Forecast product demand** – Input historical order information to predict future order quantities.
- **Media intelligence** – Maximize the value of media content by adding machine learning to media workflows such as search and discovery, content localization, compliance, monetization, and more.
- **Personalize application content** – Predict which items a user will be most interested in, and retrieve these predictions from your application in real-time.
- **Predict user activity** – Analyze user behavior to customize your website and provide a better user experience.
- **Listen to social media** – Ingest and analyze social media feeds that potentially impact business decisions.
- **Intelligent contact center** – Enhance your customer service experience and reduce costs by integrating ML into your contact center.
- **Intelligent search** – Boost business productivity and customer satisfaction by delivering accurate and useful information faster from siloed and unstructured information sources across the organization.

Cost model

With Amazon Machine Learning services, you pay only for what you use. There are no minimum fees and no upfront commitments.

AWS pre-trained AI Services cost model varies depending upon the AI service you are planning to integrate with your applications. For details, see pricing details of the respective AI services:

- Amazon Comprehend
- Amazon Forecast
- Amazon Fraud Detector
- Amazon Translate
- Amazon CodeGuru
- Amazon Textract
- Amazon Rekognition

- Amazon Polly
- Amazon Lex
- Amazon Kendra
- Amazon Personalize
- Amazon Transcribe

With Amazon SageMaker AI, you have two choices to pay, and you only pay for what you use:

- **On-demand pricing** is billed by the second, with no minimum fees and no upfront commitments.
- **SageMaker Savings Plans** offer a flexible, usage-based pricing model in exchange for a commitment to a consistent amount of usage. For details, see Amazon SageMaker AI pricing.

The ML Frameworks and Infrastructure level is intended for expert ML practitioners. ML training and inference workloads can exhibit characteristics that are steady state (such as hourly batch tagging of photos for a large population), spiky (such as kicking off new training jobs or search recommendations during promotional periods), or both. AWS has pricing options and solutions to help you optimize your infrastructure performance and costs.

For details, see the AWS Machine Learning Infrastructure.

Performance

The time it takes to create models and to request predictions from ML models depends on the number of input data records, and the types and distribution of attributes that you specify. There are a number of principles designed to help increase performance specifically for ML workloads:

- **Optimize compute for your ML workload** — Most ML workloads are very compute-intensive, because large amounts of vector multiplications and additions need to be performed on a multitude of data and parameters. Especially in Deep Learning, there is a need to scale to chipsets that provide larger queue depth, higher Arithmetic Logic Units and Register counts, to allow for massively parallel processing. Because of that, GPUs are the preferred processor type to train a Deep Learning model.
- **Define latency and network bandwidth performance requirements for your models** — Some of your ML applications might require near-instantaneous inference results to satisfy your business requirements. Offering the lowest latency possible may require the removal of

costly round trips to the nearest API endpoints. This reduction in latency can be achieved by running the inference directly on the device itself. This is known as *Machine Learning at the Edge*. A common use case for such a requirement is predictive maintenance in factories. This form of low latency and near-real-time inference at the edge allows for early indications of failure, potentially mitigating costly repairs of machinery before the failure actually happens.

- **Continuously monitor and measure system performance** — The practice of identifying and regularly collecting key metrics related to the building, training, hosting, and running predictions against a model ensures that you are able to continuously monitor the holistic success across key evaluation criteria. To validate system level resources used to support the phases of ML workloads, it's key to continuously collect and monitor system level resources such as compute, memory, and network. Requirements for ML workloads change in different phases, as training jobs are more memory intensive, while inference jobs are more compute intensive.

Durability and availability

There are key principles designed to help increase availability and durability specifically for ML workloads:

- **Manage changes to model inputs through automation** — ML workloads have additional requirements to manage changes to the data that is used to train a model to be able to recreate the exact version of a model in the event of failure or human error. Managing versions and changes through automation provides for a reliable and consistent recovery method.
- **Train once and deploy across environments** — When deploying the same version of an ML model across multiple accounts or environments, the same practice of build once that is applied to application code should be applied for model training. A specific version of a model should only be trained once and the output model artifacts should be utilized to deploy across multiple environments to avoid bringing in any unexpected changes to the model across environments.

Scalability and elasticity

There are key principles designed to help increase availability and durability specifically for ML workloads:

- **Identify the end-to-end architecture and operational model early** — Early in the ML development lifecycle, identify the end-to-end architecture and operational model for model training and hosting. This allows for early identification of architectural and operational

considerations that will be required for the development, deployment, management and integration of ML workloads.

- **Version machine learning inputs and artifacts** — Versioned inputs and artifacts enable you to recreate artifacts for previous versions of your ML workload. Version inputs are used to create models, including training data and training source code, and model artifacts.
- **Automate machine learning deployment pipelines** — Minimize human touch points in ML deployment pipelines to ensure that ML models are consistently and repeatedly deployed using a pipeline that defines how models move from development to production. Identify and implement a deployment strategy that satisfies the requirements of your use case and business problem. If required, include human quality gates in your pipeline to have humans evaluate if a model is ready to deploy to a target environment.

Interfaces

Creating a data source is as simple as adding your data to S3. To ingest data, you can use AWS Direct Connect to privately connect your data center directly to an AWS Region. To physically transfer petabytes of data in batches, use AWS Snowball Edge, or, if you have exabytes of data, use AWS Snowmobile. You can integrate your existing on-premises storage using Storage Gateway, or add cloud capabilities using AWS Snowball Edge Edge. Use Amazon Data Firehose to collect and ingest multiple streaming-data sources.

Anti-patterns

Amazon ML has the following anti-patterns:

- **Big data processing** – Data processing activities are well suited for tools like Apache Spark, which provide SQL support for data discovery, among other useful utilities. On AWS, Amazon EMR facilitates the management of Spark clusters, and enables capabilities like elastic scaling while minimizing costs through Spot Instance pricing.
- **Real-time analytics** – Collecting, processing, and analyzing the streaming data to respond in real time are well suited for tools like Kafka. On AWS, Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information and Amazon MSK is a fully managed service that makes it easy for you to build and run applications that use Apache Kafka to process streaming data. Apache Kafka is an open-source platform for building real-time streaming data pipelines and applications.

Amazon DynamoDB

Amazon DynamoDB is a fast, fully-managed NoSQL database service that makes it simple and cost effective to store and retrieve any amount of data, and serve any level of request traffic. DynamoDB helps offload the administrative burden of operating and scaling a highly-available distributed database cluster. This storage alternative meets the latency and throughput requirements of highly demanding applications by providing single-digit millisecond latency and predictable performance with seamless throughput and storage scalability.

DynamoDB stores structured data in tables, indexed by primary key, and allows low-latency read and write access to items ranging from 1 byte up to 400 KB. DynamoDB supports three data types (number, string, and binary), in both scalar and multi-valued sets. It supports document stores such as JSON, XML, or HTML in these data types. Tables do not have a fixed schema, so each data item can have a different number of attributes. The primary key can either be a single-attribute hash key or a composite hash-range key.

DynamoDB offers both global and local secondary indexes provide additional flexibility for querying against attributes other than the primary key. DynamoDB provides both eventually-consistent reads (by default), and strongly-consistent reads (optional), as well as implicit item-level transactions for item put, update, delete, conditional operations, and increment/decrement.

With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic. You can scale up or scale down your tables' throughput capacity without downtime or performance degradation.

DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs. You can create on-demand backups and enable point-in-time recovery for your Amazon DynamoDB tables. Point-in-time recovery helps protect your tables from accidental write or delete operations. With point-in-time recovery, you can restore a table to any point in time during the last 35 days.

DynamoDB enables you to delete expired items from tables automatically to help you reduce storage usage and the cost of storing data that is no longer relevant. For more information, see Expiring Items By Using DynamoDB Time to Live (TTL).

DynamoDB is integrated with other services, such as Amazon EMR, Amazon Redshift, AWS Data Pipeline, and S3 for analytics, data warehouse, data import/export, backup, and archive.

Ideal usage patterns

DynamoDB is ideal for existing or new applications that need a flexible NoSQL database with low read and write latencies, and the ability to scale storage and throughput up or down as needed without code changes or downtime.

Common use cases include:

- Mobile apps
- Gaming
- Digital ad serving
- Live voting
- Audience interaction for live events
- Sensor networks
- Log ingestion
- Access control for web-based content
- Metadata storage for Amazon S3 objects
- Ecommerce shopping carts
- Web session management

Many of these use cases require a highly available and scalable database because downtime or performance degradation has an immediate negative impact on an organization's business.

Cost model

With DynamoDB, you pay only for what you use and there is no minimum fee.

DynamoDB charges for reading, writing, and storing data in your DynamoDB tables, along with any optional features you choose to enable. DynamoDB has two capacity modes and those come with specific billing options for processing reads and writes on your tables: *on-demand* and *provisioned*.

- With **on-demand** capacity mode, DynamoDB charges you for the data reads and writes your application performs on your tables. You do not need to specify how much read and write throughput you expect your application to perform, because DynamoDB instantly accommodates your workloads as they ramp up or down.

- With **provisioned** capacity mode, you specify the number of reads and writes per second that you expect your application to require. You can use auto scaling to automatically adjust your table's capacity based on the specified utilization rate to ensure application performance while reducing costs.

New customers can start using DynamoDB for free as part of the AWS Free Usage Tier. For more information, see Amazon DynamoDB pricing.

Performance

DynamoDB is a key-value and document database that can support tables of virtually any size with horizontal scaling. This enables DynamoDB to scale to more than 500,000 requests per second for hundreds of customers.

DynamoDB supports both key-value and document data models. This enables DynamoDB to have a flexible schema, so each row can have any number of columns at any point in time. This enables you to easily adapt the tables as your business requirements change, without having to redefine the table schema as you would in relational databases.

DynamoDB Accelerator (DAX) is an in-memory cache that delivers fast read performance for your tables at scale by enabling you to use a fully managed in-memory cache. Using DAX, you can improve the read performance of your DynamoDB tables by up to ten times—taking the time required for reads from milliseconds to microseconds, even at millions of requests per second.

DynamoDB global tables replicate your data automatically across your choice of AWS Regions, and automatically scale capacity to accommodate your workloads. With global tables, your globally distributed applications can access data locally in the selected Regions to get single-digit millisecond read and write performance.

Amazon Kinesis Data Streams for DynamoDB captures item-level changes in your DynamoDB tables as a Kinesis data stream. This feature enables you to build advanced streaming applications such as real-time log aggregation, real-time business analytics, and IoT data capture. Through Kinesis Data Streams, you also can use Amazon Data Firehose to deliver DynamoDB data automatically to other AWS services.

AWS Glue Elastic Views supports DynamoDB as a source to combine and replicate data continuously across multiple databases in near-real-time.

Durability and availability

DynamoDB automatically spreads the data and traffic for your tables over a sufficient number of servers to handle your throughput and storage requirements, while maintaining consistent and fast performance. All of your data is stored on solid-state disks (SSDs) and is automatically replicated across multiple Availability Zones in an AWS Region, providing built-in high availability and data durability. You can use global tables to keep DynamoDB tables in sync across AWS Regions.

Amazon DynamoDB Streams captures all data activity that happens on your table and enables you to set up Regional replication from one geographic Region to another to provide even greater availability.

Scalability and elasticity

DynamoDB is both highly scalable and elastic. There is no limit to the amount of data that you can store in a DynamoDB table, and the service automatically allocates more storage as you store more data using the DynamoDB write API operations. Data is automatically partitioned and re-partitioned as needed, while the use of SSDs provides predictable low-latency response times at any scale. The service is also elastic, in that you can simply “dial-up” or “dial-down” the read and write capacity of a table as your needs change.

When you create a DynamoDB table, auto scaling is the default capacity setting, but you can also enable auto scaling on any table that does not have it active. Behind the scenes, DynamoDB auto scaling uses a scaling policy in Application Auto Scaling. To configure auto scaling in DynamoDB, you set the minimum and maximum levels of read and write capacity in addition to the target utilization percentage. Auto scaling uses Amazon CloudWatch to monitor a table’s read and write capacity metrics by creating CloudWatch alarms that track consumed capacity.

Interfaces

DynamoDB provides a low-level RESTAPI, as well as higher-level SDKs for Java, ET, and PHP that wrap the low-level REST API and provide some object-relational mapping (ORM) functions. These APIs provide both a management and data interface for DynamoDB. The API currently offers operations that enable table management (creating, listing, deleting, and obtaining metadata) and working with attributes (getting, writing, and deleting attributes; query using an index, and full scan).

While standard SQL isn't available, you can use the DynamoDB select operation to create SQL-like queries that retrieve a set of attributes based on criteria that you provide. You can also work with DynamoDB using the console.

Anti-patterns

DynamoDB has the following anti-patterns:

- **Prewritten application tied to a traditional relational database** – If you are attempting to port an existing application to the AWS Cloud and need to continue using a relational database, you can use either Amazon RDS (Amazon Aurora, MySQL, PostgreSQL, Oracle, or SQL Server), or one of the many pre-configured Amazon EC2 database AMIs. You can also install your choice of database software on an EC2 instance that you manage.
- **Joins or complex transactions** – While many solutions are able to leverage DynamoDB to support their users, it's possible that your application may require joins, complex transactions, and other relational infrastructure provided by traditional database platforms. If this is the case, you may want to explore Amazon Redshift, Amazon RDS, or Amazon EC2 with a self-managed database.
- **Binary large objects (BLOB) data** – If you plan on storing large (greater than 400 KB) BLOB data, such as digital video, images, or music, you'll want to consider Amazon S3. However, DynamoDB can be used in this scenario for keeping track of metadata (such as item name, size, date created, owner, location, and so on) about your binary objects.
- **Large data with low I/O rate** – DynamoDB uses SSD drives and is optimized for workloads with a high I/O rate per GB stored. If you plan to store very large amounts of data that are infrequently accessed, other storage options may be a better choice, such as Amazon S3.

Amazon Redshift

Amazon Redshift is a fast, fully-managed, petabyte-scale data warehouse service that makes it simple and cost-effective to analyze all your data efficiently using your existing business intelligence tools. It is optimized for data sets ranging from a few hundred gigabytes to a petabyte or more, and is designed to cost less than a tenth of the cost of most traditional data warehousing solutions.

Amazon Redshift delivers fast query and I/O performance for virtually any size dataset by using columnar storage technology while parallelizing and distributing queries across multiple nodes. It automates most of the common administrative tasks associated with provisioning, configuring,

monitoring, backing up, and securing a data warehouse, making it easy and inexpensive to manage and maintain. This automation enables you to build petabyte-scale data warehouses in minutes instead of weeks or months.

Amazon Redshift Spectrum enables you to run queries against exabytes of unstructured data in Amazon S3, with no loading or ETL required. When you issue a query, it goes to the Amazon Redshift SQL endpoint, which generates and optimizes a query plan. Amazon Redshift determines what data is local and what is in S3, generates a plan to minimize the amount of S3 data that needs to be read, and then requests Redshift Spectrum workers out of a shared resource pool to read and process the data from S3.

With the federated query feature in Amazon Redshift, you can query and analyze data across operational databases, data warehouses, and data lakes. It also enables you to integrate queries from Amazon Redshift on live data in external databases with queries across your Amazon Redshift and S3 environments. With cross-database queries, you can query data from any database in the Amazon Redshift cluster, regardless of which database you are connected to. Cross-database queries eliminate data copies and simplify your data organization to support multiple business groups from the same data warehouse.

AQUA (Advanced Query Accelerator) is a new distributed and hardware-accelerated cache that enables Amazon Redshift to run up to ten times faster than other enterprise cloud data warehouses by automatically boosting certain types of queries. AQUA is included with certain node types in the Amazon Redshift RA3 cluster.

There is also a feature called **Data sharing** which enables you to share data across Amazon Redshift clusters without needing to manually copy or move it for read purposes. You can have live access to data, and users can see the most up-to-date and consistent information as it's updated in the Amazon Redshift cluster.

Ideal usage patterns

Amazon Redshift is ideal for online analytical processing (OLAP) using your existing business intelligence tools. Organizations are using Amazon Redshift to:

- Analyze global sales data for multiple products
- Store historical stock trade data
- Analyze ad impressions and clicks
- Aggregate gaming data

- Analyze social trends
- Measure clinical quality, operation efficiency, and financial performance in health care
- Analyze data across the data lake (S3) and Amazon Redshift.

Cost model

An Amazon Redshift data warehouse cluster requires no long-term commitments or upfront costs. This frees you from the capital expense and complexity of planning and purchasing data warehouse capacity ahead of your needs. Charges are based on the size and number of nodes of your cluster.

There is no additional charge for backup storage up to 100% of your provisioned storage. For example, if you have an active cluster with 2 XL nodes for a total of 4 TB of storage, AWS provides up to 4 TB of backup storage on Amazon S3 at no additional charge. Backup storage beyond the provisioned storage size, and backups stored after your cluster is terminated, are billed at standard Amazon S3 rates. There is no data transfer charge for communication between S3 and Amazon Redshift.

For more information, see Amazon Redshift pricing.

Performance

Amazon Redshift uses a variety of innovations to obtain very high performance on data sets ranging in size from hundreds of gigabytes to a petabyte or more. It uses columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries.

Amazon Redshift has a massively parallel processing (MPP) architecture, parallelizing and distributing SQL operations to take advantage of all available resources. The underlying hardware is designed for high performance data processing, using local attached storage to maximize throughput between the CPUs and drives, and a 10 GigE mesh network to maximize throughput between nodes. Performance can be tuned based on your data warehousing needs: AWS offers Dense Compute (DC) with SSD drives as well as Dense Storage (DS) options.

Durability and availability

Amazon Redshift automatically detects and replaces a failed node in your data warehouse cluster. The data warehouse cluster is read-only until a replacement node is provisioned and added to

the DB, which typically only takes a few minutes. Amazon Redshift makes your replacement node available immediately and streams your most frequently accessed data from S3 first, to allow you to resume querying your data as quickly as possible.

Additionally, your data warehouse cluster remains available in the event of a drive failure; because Amazon Redshift mirrors your data across the cluster, it uses the data from another node to rebuild failed drives. Amazon Redshift clusters reside within one Availability Zone, but if you wish to have a multi-AZ set up for Amazon Redshift, you can set up a mirror and then self-manage replication and failover.

Scalability and elasticity

With a few clicks in the console or an API call, you can change the number, or type, of nodes in your data warehouse as your performance or capacity needs change. Amazon Redshift enables you to start with a single 160 GB node and scale up to a petabyte or more of compressed user data using many nodes. For more information, see Clusters and Nodes in Amazon Redshift in the *Amazon Redshift Management Guide*.

While resizing, Amazon Redshift places your existing cluster into read-only mode, provisions a new cluster of your chosen size, and then copies data from your old cluster to your new one in parallel. During this process, you pay only for the active Amazon Redshift cluster. You can continue running queries against your old cluster while the new one is being provisioned. After your data has been copied to your new cluster, Amazon Redshift automatically redirects queries to your new cluster and removes the old cluster.

Interfaces

Amazon Redshift has custom JDBC and ODBC drivers that you can download from the **Connect Client** tab of the console, which enables you to use a wide range of familiar SQL clients. You can also use standard PostgreSQL JDBC and ODBC drivers. For more information about Amazon Redshift drivers, see Amazon Redshift and PostgreSQL.

There are numerous examples of validated integrations with many popular BI and ETL vendors. Loads and unloads are attempted in parallel into each compute node to maximize the rate at which you can ingest data into your data warehouse cluster as well as to and from S3 and DynamoDB. You can easily load streaming data into Amazon Redshift using Amazon Data Firehose, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today. Metrics for compute utilization, memory utilization, storage utilization, and read/write

traffic to your Amazon Redshift data warehouse cluster are available free of charge via the console or CloudWatch API operations.

Anti-patterns

Amazon Redshift has the following anti-patterns:

- **Small datasets** – Amazon Redshift is built for parallel processing across a cluster. If your data set is less than a hundred gigabytes, you won't get all the benefits that Amazon Redshift has to offer, and Amazon RDS may be a better solution.
- **Online transaction processing (OLTP)** – Amazon Redshift is designed for data warehouse workloads producing extremely fast and inexpensive analytic capabilities. If you require a fast transactional system, you may want to choose a traditional relational database system built on Amazon RDS or a NoSQL database offering, such as DynamoDB.
- **Unstructured data** – Data in Amazon Redshift must be structured by a defined schema, rather than supporting arbitrary schema structure for each row. If your data is unstructured, you can perform extract, transform, and load (ETL) on Amazon EMR to get the data ready for loading into Amazon Redshift.
- **BLOB data** – If you plan on storing large binary files (such as digital video, images, or music), you may want to consider storing the data in S3 and referencing its location in Amazon Redshift. In this scenario, Amazon Redshift keeps track of metadata (such as item name, size, date created, owner, location, and so on) about your binary objects, but the large objects themselves are stored in S3.

Amazon OpenSearch Service

Amazon OpenSearch Service (OpenSearch Service) makes it easy to deploy, operate, and scale OpenSearch Service for log analytics, full text search, application monitoring, and more. OpenSearch Service is a fully managed service that delivers the OpenSearch Service easy-to-use APIs and real-time capabilities along with the availability, scalability, and security required by production workloads. The service offers built-in integrations with OpenSearch Dashboards, Logstash, and AWS services including Amazon Data Firehose, AWS Lambda, and Amazon CloudWatch so that you can go from raw data to actionable insights quickly.

It's easy to get started with OpenSearch Service. You can set up and configure your OpenSearch Service domain in minutes from the AWS Management Console. If you prefer programmatic access, you can use the AWS CLI or the AWS SDKs. OpenSearch Service provisions all the resources for your

domain and launches it. The service automatically detects and replaces failed OpenSearch Service nodes, reducing the overhead associated with self-managed infrastructure and OpenSearch Service software.

OpenSearch Service enables you to scale your cluster through a single API call, or a few clicks in the console. With OpenSearch Service, you get direct access to the OpenSearch Service open-source API, so that code and applications you're already using with your existing OpenSearch Service environments will work seamlessly.

In addition to X86 based instances, Amazon OpenSearch Service offers instances from the Graviton2 instance family. The instance family includes general purpose, compute-optimized, and memory-optimized instances for you to choose from. OpenSearch Service Service Graviton2 instances support OpenSearch Service versions 7.9 and above.

Ideal usage patterns

OpenSearch Service is ideal for querying and searching large amounts of data. Organizations can use OpenSearch Service to do the following:

- Analyze activity logs, such logs for customer facing applications or websites
- Analyze CloudWatch Logs with OpenSearch Service
- Analyze product usage data coming from various services and systems
- Analyze social media sentiments, CRM data, and find trends for your brand and products
- Analyze data stream updates from other AWS services, such as Amazon Kinesis Data Streams and Amazon DynamoDB
- Utilize a rich search and navigation experience.
- Monitor usage for mobile applications

Cost model

With Amazon OpenSearch Service, you pay only for what you use. There are no minimum fees or upfront commitments. You are charged for OpenSearch Service instance hours, Amazon EBS storage (if you choose this option), and standard data transfer fees.

You can get started with our free tier, which provides free usage of up to 750 hours per month of a single-AZ `t2.micro.elasticsearch` or `t2.small.elasticsearch` instance and 10 GB per month of optional Amazon EBS storage (Magnetic or General Purpose). With OpenSearch Service

Reserved Instances, you can reserve instances for a one- or three-year term and get significant savings on usage costs compared to On-Demand Instances.

OpenSearch Service enables you to add data durability through automated and manual snapshots of your cluster. OpenSearch Service provides storage space for automated snapshots free of charge for each Amazon OpenSearch Service domain. Automated snapshots are retained for a period of 14 days. Manual snapshots are charged according to Amazon S3 storage rates. Data transfer for using the snapshots is free of charge. At the time of this writing, with Graviton2 instances for OpenSearch Service, you can realize up to 44% price/performance improvement over previous generation instances.

For more information, see Amazon OpenSearch Service Pricing.

UltraWarm is a tier for OpenSearch Service that provides a cost-effective way to store large amounts of read-only data on OpenSearch Service. Rather than attached storage, UltraWarm nodes use S3 and a sophisticated caching solution to improve performance. For indexes that you are not actively writing to, query less frequently, and don't need the same performance from, UltraWarm offers significantly lower costs per GiB of data. Because warm indices are read-only unless you return them to hot storage, UltraWarm is best-suited to immutable data, such as logs.

Performance

Performance of OpenSearch Service depends on multiple factors including instance type, workload, index, number of shards used, read replicas, and storage configurations – instance storage or EBS storage (general purpose SSD). Indexes are made up of shards of data which can be distributed on different instances in multiple Availability Zones.

Read replica of the shards are maintained by OpenSearch Service in a different Availability Zone if zone awareness is checked. OpenSearch Service can use either the fast SSD instance storage for storing indexes or multiple EBS volumes. A search engine makes heavy use of storage devices and making disks faster will result in faster query and search performance. Leveraging Graviton2 instances can also improve indexing throughput, indexing latency reduction, and query performance, in comparison with the corresponding x86-based instances from the current generation.

Durability and availability

You can configure your OpenSearch Service domains for high availability by enabling the Zone Awareness option either at domain creation time or by modifying a live domain. When Zone

Awareness is enabled, OpenSearch Service distributes the instances that support the domain across two different Availability Zones. Then, if you enable replicas in OpenSearch Service, the instances are automatically distributed in such a way as to deliver cross-zone replication. You can build data durability for your OpenSearch Service domain through automated and manual snapshots.

You can use snapshots to recover your domain with preloaded data or to create a new domain with preloaded data. Snapshots are stored in Amazon S3, which is a secure, durable, highly-scalable object storage. By default, S3 automatically creates daily snapshots of each domain. In addition, you can use the S3 snapshot APIs to create additional manual snapshots. The manual snapshots are stored in S3. Manual snapshots can be used for cross-Region disaster recovery and to provide additional durability.

Scalability and elasticity

You can add or remove instances, and easily modify Amazon EBS volumes to accommodate data growth. You can write a few lines of code that will monitor the state of your domain through Amazon CloudWatch metrics and call the OpenSearch Service API to scale your domain up or down based on thresholds you set. The service will run the scaling without any downtime. OpenSearch Service supports 1 EBS volume (max size of 1.5 TB) per instance associated with a domain. With the default maximum of 20 data nodes allowed per OpenSearch Service domain, you can allocate about 30 TB of EBS storage to a single domain. You can request a service limit increase up to 100 instances per domain by creating a case with the AWS Support Center (sign-in required) With 100 instances, you can allocate about 150 TB of EBS storage to a single domain.

UltraWarm enables you to dramatically extend your data retention period and reduce costs by up to 90% over hot storage. Best of all, the interactive analytics experience remains. Query your warm indexes just like any other index, or use them to build OpenSearch Dashboards. UltraWarm uses a combination of S3 and nodes powered by the AWS Nitro System to provide a hot-like experience for aggregations and visualizations.

Cold storage lets you store any amount of infrequently accessed or historical data on your OpenSearch Service domain and analyze it on demand, at a lower cost than other storage tiers. Cold storage is appropriate if you need to do periodic research or forensic analysis on your older data. Practical examples of data suitable for cold storage include infrequently accessed logs, data that must be preserved to meet compliance requirements, or logs that have historical value.

Similar to UltraWarm storage, cold storage is backed by S3. When you need to query cold data, you can selectively attach it to existing UltraWarm nodes. You can manage the migration and lifecycle of your cold data manually or with Index State Management policies.

Interfaces

OpenSearch Service supports many of the commonly used OpenSearch APIs, so code, applications, and popular tools that you're already using with your current OpenSearch environments will work seamlessly. For a full list of supported OpenSearch operations, see OpenSearch Service documentation .

The AWS CLI, API, or the AWS Management Console can be used for creating and managing your domains as well.

OpenSearch Service supports integration with several AWS services, including streaming data from S3 buckets, Amazon Kinesis Data Streams, and DynamoDB Streams. Both integrations use a Lambda function as an event handler in the cloud that responds to new data in Amazon S3 and Amazon Kinesis Data Streams by processing it and streaming the data to your OpenSearch Service domain. OpenSearch Service also integrates with Amazon CloudWatch for monitoring OpenSearch Service domain metrics and CloudTrail for auditing configuration API calls to OpenSearch Service domains.

OpenSearch Service includes built-in integration with OpenSearch Dashboards, an open-source analytics and visualization platform and supports integration with Logstash, an open-source data pipeline that helps you process logs and other event data. You can set up your OpenSearch Service domain as the backend store for all logs coming through your Logstash implementation to easily ingest structured and unstructured data from a variety of sources.

Fine-grained access control

Fine-grained access control offers additional ways of controlling access to your data on OpenSearch Service. For example, depending on who makes the request, you might want a search to return results from only one index. You might want to hide certain fields in your documents or exclude certain documents altogether. Fine-grained access control offers the following benefits:

- Role-based access control
- Security at the index, document, and field level
- OpenSearch Dashboards multi-tenancy
- HTTP basic authentication for OpenSearch Service and OpenSearch Dashboards

SAML authentication for OpenSearch Dashboards lets you use your existing identity provider to offer single sign-on (SSO) for OpenSearch Dashboards on OpenSearch Service domains running

OpenSearch Service 6.7 or later. To use SAML authentication, you must enable fine-grained access control.

Rather than authenticating through Amazon Cognito or the internal user database, SAML authentication for OpenSearch Dashboards lets you use third-party identity providers to log in to OpenSearch Dashboards, manage fine-grained access control, search your data, and build visualizations. OpenSearch Service supports providers that use the SAML 2.0 standard, such as Okta, Keycloak, Active Directory Federation Services (AD FS), and Auth0. Requests from OpenSearch Service to third-party providers aren't explicitly encrypted with a service provider certificate.

SAML authentication for OpenSearch Dashboards is only for accessing OpenSearch Dashboards through a web browser. Your SAML credentials do not let you make direct HTTP requests to the OpenSearch Service or OpenSearch Dashboards APIs.

Anti-patterns

Amazon OpenSearch Service has the following anti-patterns:

- **Online transaction processing (OLTP)** - OpenSearch Service is a real-time distributed search and analytics engine. There is no support for transactions or processing on data manipulation. If your requirement is for a fast transactional system, then a traditional relational database system built on Amazon RDS, or a NoSQL database offering functionality such as DynamoDB, is a better choice.
- **Ad hoc data querying** – While OpenSearch Service takes care of the operational overhead of building a highly scalable OpenSearch Service cluster, if running ad hoc queries or one-off queries against your data set is your use-case, Amazon Athena is a better choice. Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL, without provisioning servers.

Amazon QuickSight

Amazon QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service built for the cloud. It makes it easy for all employees within an organization to build visualizations, perform ad hoc analysis, and quickly get business insights from their data, anytime, on any device. It can connect to a wide variety of data sources including flat files such as CSV and Excel, access on-premises databases including Oracle, SQL Server, MySQL, and PostgreSQL, AWS resources such as Amazon RDS databases, Amazon Redshift, Amazon Athena,

and Amazon S3, and SaaS solutions such as Salesforce, ServiceNow, and Adobe Analytics. Amazon QuickSight enables organizations to scale their business analytics capabilities to hundreds of thousands of users, and delivers fast and responsive query performance by using a robust in-memory engine (SPICE).

Amazon QuickSight is built with "SPICE" – a Super-fast, Parallel, In-memory Calculation Engine. Built from the ground up for the cloud, SPICE uses a combination of columnar storage, in-memory technologies enabled through the latest hardware innovations, and machine code generation to run interactive queries on large datasets and get rapid responses. SPICE supports rich calculations to help you derive valuable insights from your analysis without worrying about provisioning or managing infrastructure. Data in SPICE is persisted until it is explicitly deleted by the user. SPICE also automatically replicates data for high availability and enables Amazon QuickSight to scale to hundreds of thousands of users who can all simultaneously perform fast interactive analysis across a wide variety of AWS data sources.

QuickSight leverages the AWS proven ML capabilities, making it easy for BI teams to perform advanced analytics (for example, what-if analyses, anomaly detection, ML-based forecasting, churn prediction, and so on) without prior data science experience. You can use QuickSight's pre-built models, or bring your own ML models from Amazon SageMaker AI.

Ideal usage patterns

Amazon QuickSight is an ideal Business Intelligence tool, allowing end users to create visualizations that provide insight into their data to help them make better business decisions. Amazon QuickSight can be used to do the following:

- Quick interactive ad hoc exploration and optimized visualization of data
- Create and share dashboards and KPIs to provide insight into your data
- Create *Stories*, which are guided tours through specific views of an analysis and enable you to share insights and collaborate with others. They are used to convey key points, a thought process, or the evolution of an analysis for collaboration
- Analyze and visualize data coming from logs and stored in S3
- Analyze and visualize data from on premise databases like SQL Server, Oracle, PostGreSQL, and MySQL
- Analyze and visualize data in various AWS resources such as Amazon RDS databases, Amazon Redshift, Amazon Athena, and Amazon S3.
- Analyze and visualize data in software as a service (SaaS) applications such as Salesforce.

- Analyze and visualize data in data sources that can be connected to using JDBC/ODBC connection.
- Enhance with your QuickSight dashboard with **Machine Learning Insights** and **Forecasts**
- Ask natural language questions on your data and have Amazon QuickSight Q automatically build your visualization

Cost model

Amazon QuickSight has two different editions for pricing; *Standard* edition and *Enterprise* edition.

- The **Standard Edition** is ideal for personal data analysis and exploration, it offers a full set of features for creating and sharing data visualizations, but it does not provide some of the Enterprise security and advanced options, a comparison of the features of the two editions can be found on the Amazon QuickSight Pricing page. The Standard Edition is priced at $9/user/month for an annual subscription ($12/GB/month for the month-to-month option) with ten GB of SPICE capacity included. You can get additional SPICE capacity for $.25/GB/month.
- The **Enterprise** edition delivers insights at scale and offers embedding options, advanced security (data encryption at rest, row and column level security, access on private VPC and on premises), ML insights, folders, and templates.

  It is priced at $18/author/month ($24/GB/month for the month-to-month option) with ten GB of SPICE capacity included. Readers have a per-session pricing up to $5/user/month ($.30 per session; 1 session = 30 minutes from login).

  For large-scale deployments and public embedding, there is a Session capacity pricing starting at $250 per month for 500 sessions.

Performance

Amazon QuickSight is built with SPICE. Built from the ground up for the cloud, SPICE uses a combination of columnar storage, in-memory technologies enabled through the latest hardware innovations, and machine code generation to run interactive queries on large datasets and get rapid responses.

Durability and availability

SPICE automatically replicates data for high availability and enables Amazon QuickSight to scale to hundreds of thousands of users who can all simultaneously perform fast interactive analysis across a wide variety of AWS data sources.

Scalability and elasticity

Amazon QuickSight is a fully managed service and it internally takes care of scaling to meet the demands of your end users. With Amazon QuickSight you don't need to worry about scale. You can seamlessly grow your data from a few hundred megabytes to many terabytes of data without managing any infrastructure.

Interfaces

Amazon QuickSight can connect to a wide variety of data sources including flat files (CSV, TSV, CLF, ELF), connect to on-premises databases like SQL Server, MySQL, and PostgreSQL, and AWS data sources including Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Athena and Amazon S3, and SaaS, applications like Salesforce. You can also export data from any type of chart or graph and the export contains only the data in the fields that are currently visible in the selected visualization.

You can share an analysis, dashboard, or story by choosing **Share with users and groups in your account** on the application bar from the Amazon QuickSight service interface. You can select the recipients (email address, username, or group name), permission levels, and other options, before sharing the content with others.

Anti-patterns

While Amazon QuickSight can perform some transformations, it is not a full-fledged ETL tool. AWS offers AWS Glue, which is a fully managed ETL service that makes it easy for customers to prepare and load their data for analytics.

Amazon Compute Services (EC2, EKS, and ECS)

Amazon EC2, with instances acting as AWS virtual machines, provides an ideal platform for operating your own self-managed big data analytics applications on AWS infrastructure. Almost any software you can install on Linux or Windows virtualized environments can be run on Amazon EC2, and you can use the pay-as-you-go pricing model.

AWS Graviton processors are custom built by AWS using 64-bit Arm Neoverse cores to deliver the best price performance for your cloud workloads running in Amazon EC2. Amazon EC2 provides the broadest and deepest portfolio of compute instances, including many that are powered by latest-generation Intel and AMD processors. AWS Graviton processors add even more choice to help customers optimize performance and cost for their workloads.

What you don't get are the application-level managed services that come with the other services mentioned in this whitepaper. There are many options for self-managed big data analytics:

- A NoSQL offering, such as MongoDB
- A data warehouse or columnar store like Vertica
- A Hadoop cluster
- An Apache Storm cluster
- An Apache Kafka environment

Any self-managed big data workload that runs on EC2 can also run on an AWS fully managed container orchestration service such as Amazon ECS, Amazon EKS, and AWS Fargate. Fargate is a serverless compute engine for containers that works with ECS and EKS.

Ideal usage patterns

- **Specialized environment** – When running a custom application, a variation of a standard Hadoop set or an application not covered by another AWS offering, Amazon EC2 provides the flexibility and scalability to meet your computing needs.
- **Compliance requirements** – Certain compliance requirements may require you to run applications yourself on Amazon EC2 instead of using a managed service offering.

Cost model

Amazon EC2 has a variety of instance types in a number of instance families (standard, high CPU, high memory, high I/O, and so on), and different pricing options (On-Demand, Compute Savings plan, Reserved, and Spot). At the time of this writing, when running applications on ECS, you pay only for underlying EC2 instances, with no additional charge for using ECS. However, for EKS, you pay an additional $0.10 per hour for each EKS cluster you have, along with underlying EC2 instances. AWS Fargate pricing is calculated based on the vCPU, memory, and storage resources

used from the time you start to download your container image until the Amazon ECS task or Amazon EKS2 pod finishes, rounded up to the nearest second.

While cost is dependent on various factors based on the use case, Graviton2 instances have in general been able to provide better price/performance over previous generation instances. Depending on your application requirements, you may want to use additional services along with Amazon EC2, EKS, or ECS, such as Amazon Elastic Block Store (Amazon EBS) for directly attached persistent storage, or S3 as a durable object store; each comes with its own pricing model. If you do run your big data application on Amazon EC2, EKS, or ECS, you are responsible for any license fees just as you would be in your own data center. The AWS Marketplace offers many different third-party, big data software packages pre-configured to launch with a simple click of a button.

Performance

Performance in Amazon EC2, EKS, or ECS is driven by the instance type that you choose for your big data platform. Each instance type has a different amount of CPU, RAM, storage, IOPs, and networking capability so that you can pick the right performance level for your application requirements.

Durability and availability

Critical applications should be run in a cluster across multiple Availability Zones within an AWS Region so that any instance or data center failure does not affect application users. For non-uptime critical applications, you can back up your application to Amazon S3 and restore to any Availability Zone in the Region if an instance or zone failure occurs. Other options exist, depending on which application you are running and the requirements, such as mirroring your application.

Scalability and elasticity

Auto Scaling is a service that enables you to automatically scale your Amazon EC2 capacity up or down according to conditions that you define. With Auto Scaling, you can ensure that the number of EC2 instances you're using scales up seamlessly during demand spikes to maintain performance, and scales down automatically during demand lulls to minimize costs. Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Auto Scaling is enabled by CloudWatch and available at no additional charge beyond CloudWatch fees.

Interfaces

Amazon EC2, EKS, and ECS can be managed programmatically via API, SDK, or the AWS Management Console. Metrics for compute utilization, memory utilization, storage utilization, network consumption, and read/write traffic to your instances are free of charge using the console or CloudWatch API operations.

The interfaces for your big data analytics software that you run on top of Amazon EC2 varies based on the characteristics of the software you choose.

Anti-patterns

Amazon EC2 has the following anti-patterns:

- **Managed service** – If your requirement is a managed service offering where you abstract the infrastructure layer and administration from the big data analytics, then this "do it yourself" model of managing your own analytics software on Amazon EC2 may not be the correct choice.
- **Lack of expertise or resources** – If your organization does not have, or does not want to expend, the resources or expertise to install and manage a high-availability installation for the system in question, you should consider using the AWS equivalent such as Amazon EMR, DynamoDB, Amazon Kinesis Data Streams, or Amazon Redshift.

Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to setup or manage, and you can start analyzing data immediately. You don't need to load your data into Athena, as it works directly with data stored in S3. Just log into the Athena Console, define your table schema, and start querying. Amazon Athena uses Presto with full ANSI SQL support and works with a variety of standard data formats, including CSV, JSON, ORC, Apache Parquet, and Apache Avro.

Ideal usage patterns

- **Interactive ad hoc querying for web logs** — Athena is a good tool for interactive one-time SQL queries against data on Amazon S3. For example, you could use Athena to run a query on web and application logs to troubleshoot a performance issue. You simply define a table for your data and start querying using standard SQL. Athena integrates with Amazon QuickSight for easy visualization.

- **To query staging data before loading into Redshift** — You can stage your raw data in S3 before processing and loading it into Amazon Redshift, and then use Athena to query that data.
- **Send AWS service logs to S3 for analysis with Athena —** CloudTrail, CloudFront, ELB/ALB and VPC flow logs can be analyzed with Athena. AWS CloudTrail logs include details about any API calls made to your AWS services, including from the console. CloudFront logs can be used to explore users' surfing patterns across web properties served by CloudFront. Querying ELB/ALB logs allows you to see the source of traffic, latency, and bytes transferred to and from Elastic Load Balancing instances and backend applications. VPC flow logs capture information about the IP traffic going to and from network interfaces in VPCs in the Amazon Virtual Private Cloud (Amazon VPC). The logs enable you to investigate network traffic patterns and identify threats and risks across your VPC estate.
- **Building Interactive Analytical Solutions with notebook-based solutions such as RStudio, Jupyter, or Zeppelin —** Data scientists and Analysts are often concerned about managing the infrastructure behind big data platforms while running notebook-based solutions such as RStudio, Jupyter, and Zeppelin. Amazon Athena makes it easy to analyze data using standard SQL without the need to manage infrastructure. Integrating these notebook-based solutions with Amazon Athena gives data scientists a powerful platform for building interactive analytical solutions.
- **Query data in relational, non-relational, object and custom data sources leveraging Athena Federated Query** — The Amazon Athena federated query allows the user to run SQL queries across data in relational, non-relational, object and custom data sources, with options to either query the data in place or extract the data from these data sources and store it in S3.

Cost model

Amazon Athena has simple pay-as-you-go pricing, with no up-front costs or minimum fees, and you'll only pay for the resources you consume. It is priced per query, $5 per TB of data scanned, and charges based on the amount of data scanned by the query. You can save from 30% to 90% on your per-query costs and get better performance by compressing, partitioning, and converting your data into columnar formats. Converting data to the columnar format allows Athena to read only the columns it needs to process the query.

You are charged for the number of bytes scanned by Amazon Athena, rounded up to the nearest megabyte, with a 10 MB minimum per query. There are no charges for Data Definition Language (DDL) statements like `CREATE/ALTER/DROP TABLE`, statements for managing partitions, or failed queries. Canceled queries are charged based on the amount of data scanned.

Because federated queries invoke Lambda functions in your account, you are charged for Lambda when a Federated query is made.

Performance

You can improve the performance of your query by compressing, partitioning, and converting your data into columnar formats. Amazon Athena supports open source columnar data formats such as Apache Parquet and Apache ORC. Converting your data into a compressed, columnar format lowers your cost and improves query performance by enabling Athena to scan less data from S3 when running your query.

Durability and availability

Amazon Athena is highly available and executes queries using compute resources across multiple facilities, automatically routing queries appropriately if a particular facility is unreachable. Athena uses Amazon S3 as its underlying data store, making your data highly available and durable. Amazon S3 provides durable infrastructure to store important data and is designed for durability of 99.999999999% of objects. Your data is redundantly stored across multiple facilities and multiple devices in each facility.

Scalability and elasticity

Athena is serverless, so there is no infrastructure to setup or manage, and you can start analyzing data immediately. Because it is serverless it can scale automatically, as needed.

Security, authorization, and encryption

Amazon Athena allows you to control access to your data by using AWS IAM policies, Access Control Lists (ACLs), and Amazon S3 bucket policies. With IAM policies, you can grant users fine-grained control to your S3 buckets. By controlling access to data in S3, you can restrict users from querying it using Athena. You can query data that's been protected by:

- Server-side encryption with an S3-managed key
- Server-side encryption with an AWS KMS-managed key
- Client-side encryption with an AWS KMS-managed key

Amazon Athena also can directly integrate with AWS Key Management System (KMS) to encrypt your result sets.

Interfaces

Querying can be done by using the Athena Console. Athena also supports CLI, API via SDK and JDBC. Athena also integrates with Amazon QuickSight for creating visualizations based on the Athena queries.

Athena Federated Query leverages Lambda as data source connectors as its extension to make queries in sources other than S3. Sources such as Amazon CloudWatch Logs, DynamoDB, Amazon DocumentDB, Amazon RDS, JDBC-compliant Postgres, and MySQL databases are natively supported by Athena Federated Query. For others, you can use Athena Query Federation SDKs to write custom connectors.

Anti-patterns

Amazon Athena has the following anti-patterns:

- **Enterprise Reporting and Business Intelligence Workloads** – Amazon Redshift is a better tool for enterprise reporting and BI workloads involving iceberg queries or cached data at the nodes. Data warehouses pull data from many sources, format and organize it, store it, and support complex, high speed queries that produce business reports. The query engine in Amazon Redshift has been optimized to perform especially well on data warehouse workloads.
- **ETL Workloads –** You should use Amazon EMR/AWS Glue if you are looking for an ETL tool to process extremely large datasets and analyze them with the latest big data processing frameworks such as Spark, Hadoop, Presto, or Hbase.
- **RDBMS –** Athena is not a relational/transactional database. It is not meant to be a replacement for SQL engines like MySQL.

Solving big data problems on AWS

This whitepaper has examined some tools available on AWS for big data analytics. This paper provides a good reference point when starting to design your big data applications. However, there are additional aspects you should consider when selecting the right tools for your specific use case. In general, each analytical workload has certain characteristics and requirements that dictate which tool to use, such as:

- How quickly do you need analytic results: in real time, in seconds, or is an hour a more appropriate time frame?
- How much value will these analytics provide your organization and what budget constraints exist?
- How large is the data and what is its growth rate?
- How is the data structured?
- What integration capabilities do the producers and consumers have?
- How much latency is acceptable between the producers and consumers?
- What is the cost of downtime or how available and durable does the solution need to be?
- Is the analytic workload consistent or elastic?

Each one of these questions helps guide you to the right tool. In some cases, you can simply map your big data analytics workload into one of the services based on a set of requirements. However, in most real-world, big data analytic workloads, there are many different, and sometimes conflicting, characteristics and requirements on the same data set.

For example, some result sets may have real-time requirements as a user interacts with a system, while other analytics could be batched and run on a daily basis. These different requirements over the same data set should be decoupled and solved by using more than one tool. If you try to solve both of these examples using the same toolset, you end up either over-provisioning or therefore overpaying for unnecessary response time, or you have a solution that does not respond fast enough to your users in real time. Matching the best-suited tool to each analytical problem results in the most cost-effective use of your compute and storage resources.

Big data doesn't need to mean "big costs". So, when designing your applications, it's important to make sure that your design is cost efficient. If it's not, relative to the alternatives, then it's probably not the right design. Another common misconception is that using multiple tool sets to solve a big

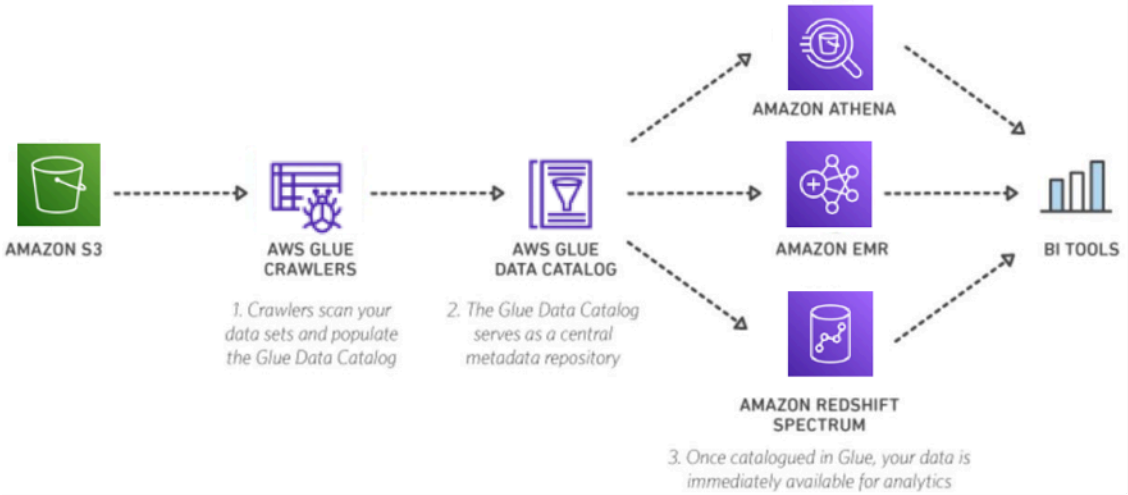
data problem is more expensive or harder to manage than using one big tool. If you take the same example of two different requirements on the same data set, the real-time request may be low on CPU but high on I/O, while the slower processing request may be very compute intensive.

Decoupling can end up being much less expensive and easier to manage, because you can build each tool to exact specifications and not overprovision. With the AWS pay-as-you-go model, this equates to a much better value because you could run the batch analytics in just one hour and therefore only pay for the compute resources for that hour. Also, you may find this approach easier to manage rather than leveraging a single system that tries to meet all of the requirements. Solving for different requirements with one tool results in attempting to fit a square peg (real-time requests) into a round hole (a large data warehouse).

The AWS platform makes it easy to decouple your architecture by having different tools analyze the same data set. AWS services have built-in integration so that moving a subset of data from one tool to another can be done very easily and quickly using parallelization. Following are some real world, big data analytics problem scenarios, and an AWS architectural solution for each.

Example 1: Queries against an Amazon S3 data lake

Data lakes are an increasingly popular way to store and analyze both structured and unstructured data. If you use an Amazon S3 data lake, AWS Glue can make all your data immediately available for analytics without moving the data. AWS Glue crawlers can scan your data lake and keep the AWS Glue Data Catalog in sync with the underlying data. You can then directly query your data lake with Amazon Athena and Amazon Redshift Spectrum. You can also use the AWS Glue Data Catalog as your external Apache Hive Metastore for big data applications running on Amazon EMR.



Queries against an Amazon S3 data lake

1. An AWS Glue crawler connects to a data store, progresses through a prioritized list of classifiers to extract the schema of your data and other statistics, and then populates the AWS Glue Data Catalog with this metadata. Crawlers can run periodically to detect the availability of new data as well as changes to existing data, including table definition changes. Crawlers automatically add new tables, new partitions to existing table, and new versions of table definitions. You can customize AWS Glue crawlers to classify your own file types.
2. The AWS Glue Data Catalog is a central repository to store structural and operational metadata for all your data assets. For a given data set, you can store its table definition, physical location, add business relevant attributes, as well as track how this data has changed over time. The AWS Glue Data Catalog is Apache Hive Metastore compatible and is a drop-in replacement for the Apache Hive Metastore for Big Data applications running on Amazon EMR. For more information on setting up your EMR cluster to use AWS Glue Data Catalog as an Apache Hive Metastore, see AWS Glue documentation.
3. The AWS Glue Data Catalog also provides out-of-box integration with Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. After you add your table definitions to the AWS Glue Data Catalog, they are available for ETL and also readily available for querying in Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum so that you can have a common view of your data between these services.
4. Using a BI tool like Amazon QuickSight enables you to easily build visualizations, perform ad hoc analysis, and quickly get business insights from your data. Amazon QuickSight supports data sources such as Amazon Athena, Amazon Redshift Spectrum, Amazon S3 and many others. See Supported Data Sources.

Example 2: Capturing and analyzing sensor data

An international air conditioner manufacturer has many large air conditioners that it sells to various commercial and industrial companies. Not only do they sell the air conditioner units but, to better position themselves against their competitors, they also offer add-on services where you can see real-time dashboards in a mobile app or a web browser. Each unit sends its sensor information for processing and analysis. This data is used by the manufacturer and its customers. With this capability, the manufacturer can visualize the dataset and spot trends.

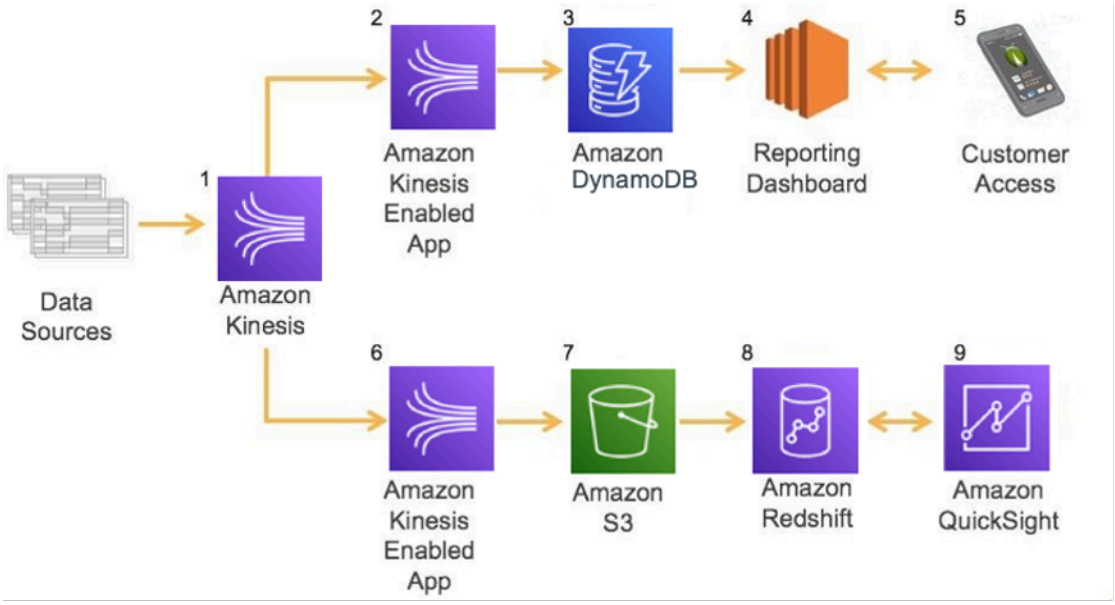
Currently, they have a few thousand pre-purchased air conditioning (A/C) units with this capability. They expect to deliver these to customers in the next couple of months and are hoping that, in time, thousands of units throughout the world will use this platform. If successful, they would like to expand this offering to their consumer line as well, with a much larger volume and a greater

market share. The solution needs to be able to handle massive amounts of data and scale as they grow their business without interruption. How should you design such a system?

First, break it up into two work streams, both originating from the same data:

- A/C unit's current information with near-real-time requirements and a large number of customers consuming this information
- All historical information on the A/C units to run trending and analytics for internal use

The data-flow architecture in the following figure shows how to solve this big data problem.



Capturing and analyzing sensor data

1. The process begins with each A/C unit providing a constant data stream to Amazon Kinesis Data Streams. This provides an elastic and durable interface the units can talk to that can be scaled seamlessly as more and more A/C units are sold and brought online.
2. Using the Amazon Kinesis Data Streams-provided tools such as the Kinesis Client Library or SDK, a simple application is built on Amazon EC2 to read data as it comes into Amazon Kinesis Data Streams, analyze it, and determine if the data warrants an update to the real-time dashboard. It looks for changes in system operation, temperature fluctuations, and any errors that the units encounter.
3. This data flow needs to occur in near real time so that customers and maintenance teams can be alerted quickly if there is an issue with the unit. The data in the dashboard does have some aggregated trend information, but it is mainly the current state as well as any system errors. So,

the data needed to populate the dashboard is relatively small. Additionally, there will be lots of potential access to this data from the following sources:

- Customers checking on their system via a mobile device or browser
- Maintenance teams checking the status of its fleet
- Data and intelligence algorithms and analytics in the reporting platform spot trends that can be then sent out as alerts, such as if the A/C fan has been running unusually long with the building temperature not going down.

  DynamoDB was chosen to store this near real-time data set because it is both highly available and scalable; throughput to this data can be easily scaled up or down to meet the needs of its consumers as the platform is adopted and usage grows.

4. The reporting dashboard is a custom web application that is built on top of this data set and run on Amazon EC2. It provides content based on the system status and trends as well as alerting customers and maintenance crews of any issues that may come up with the unit.
5. The customer accesses the data from a mobile device or a web browser to get the current status of the system and visualize historical trends.

   The data flow (steps 2-5) that was just described is built for near real-time reporting of information to human consumers. It is built and designed for low latency and can scale very quickly to meet demand. The data flow (steps 6-9) that is depicted in the lower part of the diagram does not have such stringent speed and latency requirements. This allows the architect to design a different solution stack that can hold larger amounts of data at a much smaller cost per byte of information and choose less expensive compute and storage resources.
6. To read from the Amazon Kinesis stream, there is a separate Amazon Kinesis-enabled application that probably runs on a smaller EC2 instance that scales at a slower rate. While this application is going to analyze the same data set as the upper data flow, the ultimate purpose of this data is to store it for long-term record and to host the data set in a data warehouse. This data set ends up being all data sent from the systems and allows a much broader set of analytics to be performed without the near real-time requirements.
7. The data is transformed by the Amazon Kinesis-enabled application into a format that is suitable for long-term storage, for loading into its data warehouse, and storing on Amazon S3. The data on Amazon S3 not only serves as a parallel ingestion point to Amazon Redshift, but is durable storage that will hold all data that ever runs through this system; it can be the single source of truth. It can be used to load other analytical tools if additional requirements arise. Amazon S3 also comes with native integration with Amazon Glacier, if any data needs to be cycled into long-term, low-cost storage.

8. Amazon Redshift is again used as the data warehouse for the larger data set. It can scale easily when the data set grows larger, by adding another node to the cluster.
9. For visualizing the analytics, one of the many partner visualization platforms can be used via the OBDC/JDBC connection to Amazon Redshift. This is where the reports, graphs, and ad hoc analytics can be performed on the data set to find certain variables and trends that can lead to A/C units underperforming or breaking.

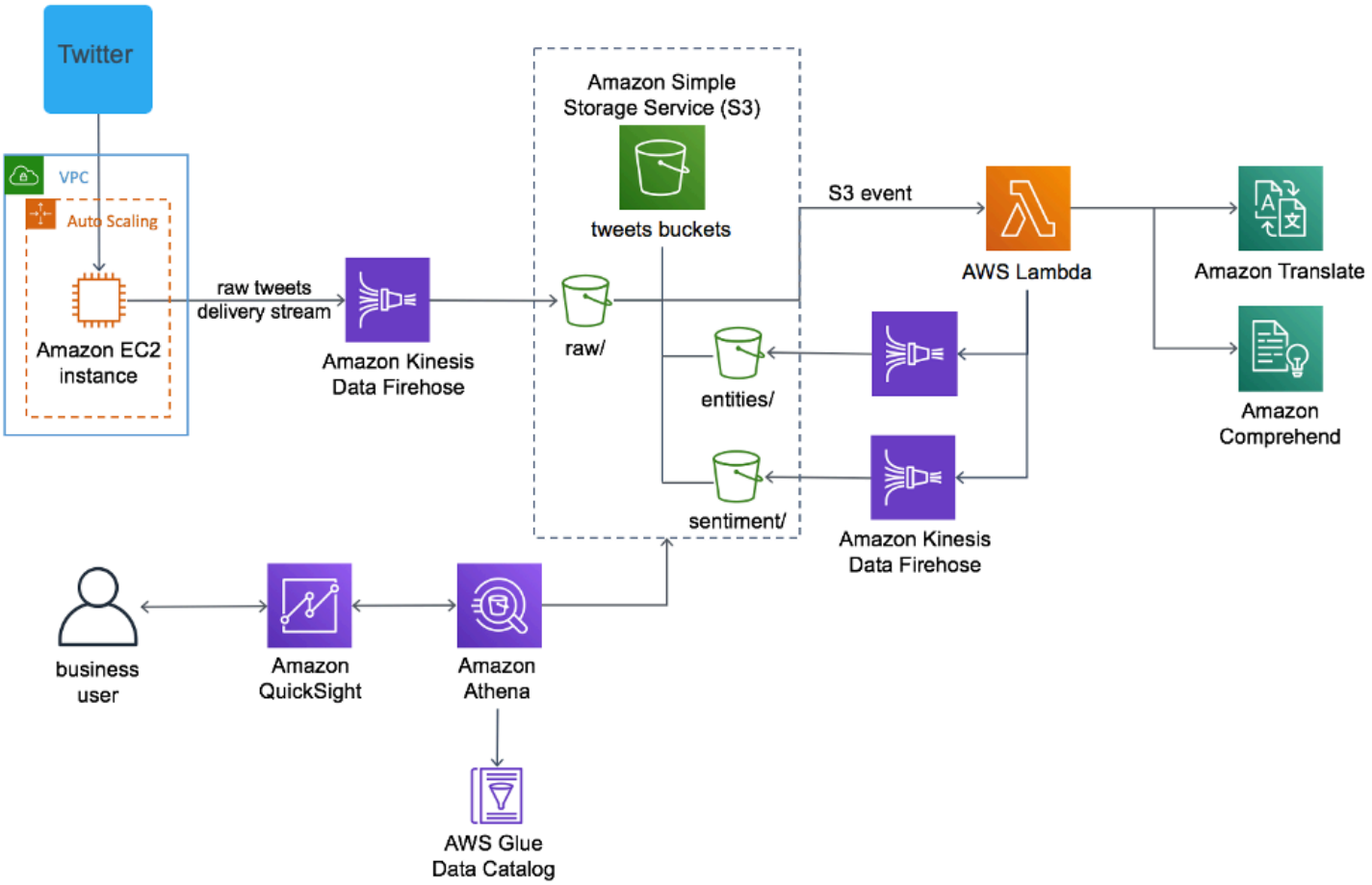
This architecture can start off small and grow as needed. Additionally, by decoupling the two different work streams from each other, they can grow at their own rate without upfront commitment, allowing the manufacturer to assess the viability of this new offering without a large initial investment. You could imagine further additions, such as adding Amazon ML to predict how long an A/C unit will last and preemptively sending out maintenance teams based on its prediction algorithms to give their customers the best possible service and experience. This level of service would be a differentiator to the competition and lead to increased future sales.

Example 3: sentiment analysis of social media

A large toy maker has been growing very quickly and expanding their product line. After each new toy release, the company wants to understand how consumers are enjoying and using their products. Additionally, the company wants to ensure that their consumers are having a good experience with their products. As the toy system grows, the company wants to ensure that their products are still relevant to their customers and that they can plan future roadmaps items based on customer feedback. The company wants to capture the following insights from social media:

- Understand how consumers are using their products
- Ensure customer satisfaction
- Plan future roadmaps

Capturing the data from various social networks is relatively easy but the challenge is building the intelligence programmatically. After the data is ingested, the company wants to be able to analyze and classify the data in a cost-effective and programmatic way. To do this, they can use the architecture in the following figure.



Sentiment analysis of social media

1. First, deploy an Amazon EC2 instance in an Amazon VPC that ingests tweets from Twitter.
2. Next, create an Amazon Data Firehose delivery stream that loads the streaming tweets into the raw prefix in the solution's S3 bucket.
3. S3 invokes a Lambda function to analyze the raw tweets using Amazon Translate to translate non-English tweets into English, and Amazon Comprehend to use natural language-processing (NLP) to perform entity extraction and sentiment analysis.
4. A second Firehose delivery stream loads the translated tweets and sentiment values into the sentiment prefix in the S3 bucket. A third delivery stream loads entities in the entities prefix in the S3 bucket.
5. This architecture also deploys a data lake that includes AWS Glue for data transformation, Amazon Athena for data analysis, and QuickSight for data visualization. AWS Glue Data Catalog contains a logical database used to organize the tables for the data in S3. Athena uses these

table definitions to query the data stored in S3 and return the information to an QuickSight dashboard.

By using ML and BI services from AWS including Amazon Translate, Amazon Comprehend, Amazon Kinesis, Amazon Athena, and Amazon QuickSight, you can build meaningful, low-cost social media dashboards to analyze customer sentiment, which can lead to better opportunities for acquiring leads, improve website traffic, strengthen customer relationships, and improve customer service.

This example solution automatically provisions and configures the AWS services necessary to capture multi-language tweets in near-real-time, translate them, and display them on a dashboard powered by Amazon QuickSight. You can also capture both the raw and enriched datasets and durably store them in the solution's data lake. This enables data analysts to quickly and easily perform new types of analytics and ML on this data. For more information, see the AI-Driven Social Media Dashboard solution.

Conclusion

As more and more data is generated and collected, data analysis requires scalable, flexible, and high performing tools to provide insights in a timely fashion. However, organizations are facing a growing big data environment, where new tools emerge and become outdated very quickly. Therefore, it can be very difficult to keep pace and choose the right tools.

This whitepaper offers a first step to help you solve this challenge. With a broad set of managed services to collect, process, and analyze big data, AWS makes it easier to build, deploy, and scale big data applications. This enables you to focus on business problems instead of updating and managing these tools.

AWS provides many solutions to address your big data analytic requirements. Most big data architecture solutions use multiple AWS tools to build a complete solution. This approach helps meet stringent business requirements in the most cost-optimized, performant, and resilient way possible. The result is a flexible big data architecture that is able to scale along with your business.

Contributors

The following individuals and organizations contributed to this document:

- Erik Swensson, Manager, Solutions Architecture, Amazon Web Services
- Erick Dame, Solutions Architect, Amazon Web Services
- Shree Kenghe, Solutions Architect, Amazon Web Services
- Changbin Gong, Solutions Architect, Amazon Web Services
- Raghavarao Sodabathina, EnterpriseSolutions Architect, Amazon Web Services
- Fabrizio Napolitano, Solutions Architect, Amazon Web Services
- Chinmayi Narasimhadevara, Solutions Architect, Amazon Web Services
- Aishwarya Subramaniam, Solutions Architect, Amazon Web Services
- Pavitra Krishnan, Solutions Architect, Amazon Web Services

Further reading

The following resources can help you get started in running big data analytics on AWS:

- Analytics on AWS — View the comprehensive portfolio of big data services as well as links to other resources such AWS big data partners, tutorials, articles, and AWS Marketplace offerings on big data solutions.
- Read the AWS Big Data Blog — The blog features real life examples and ideas updated regularly to help you collect, store, clean, process, and visualize big data.
- Try one of the Analytics Quick Starts — Explore the rich environment of products designed to address big data challenges using AWS. Test Drives are developed by AWS Partner Network (APN) Consulting and Technology partners and are provided free of charge for education, demonstration, and evaluation purposes.
- Take an AWS training course on big data — The Big Data on AWS course introduces cloud-based big data solutions and Amazon EMR. AWS shows how to use Amazon EMR to process data using the broad system of Hadoop tools such as Pig and Hive. AWS also teaches how to create big data environments, work with DynamoDB and Amazon Redshift, understand the benefits of Amazon Kinesis Streams, and leverage best practices to design big data environments for security and cost-effectiveness.
- View the Big Data Customer Case Studies — Learn from the experience of other customers who have built powerful and successful big data platforms on the AWS Cloud.

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor update	Fix non-inclusive language.	April 6, 2022
Whitepaper updated	Revised to add informati on on Amazon SageMaker AI, Amazon EMR Studio, AWS Glue DataBrew, AWS Lake Formation, and general updates throughout.	July 26, 2021
Whitepaper updated	Revised to add information on Amazon Athena, AWS QuickSight, AWS Glue, and general updates throughout.	December 1, 2018
Whitepaper updated	Revised to add information on Amazon Machine Learning, AWS Lambda, Amazon OpenSearch Service; general update.	January 1, 2016
Initial publication	Whitepaper first published.	December 1, 2014

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.